CROWD AND AI POWERED MANIPULATION: CHARACTERIZATION AND DETECTION

A Dissertation

by

PARISA KAGHAZGARAN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	James Caverlee
Committee Members,	Frank Shipman
	Ruihong Huang
	Patrick Burkart
Head of Department,	Scott Schaefer

December 2020

Major Subject: Computer Science and Engineering

ABSTRACT

User reviews are ubiquitous. They power online review aggregators that influence our daily-based decisions, from what products to purchase (e.g., Amazon), movies to view (e.g., Netflix, HBO, Hulu), restaurants to patronize (e.g., Yelp), and hotels to book (e.g., TripAdvisor, Airbnb). In addition, policy makers rely on online commenting platforms like Regulations.gov and FCC.gov as a means for citizens to voice their opinions about public policy issues. However, showcasing the opinions of fellow users has a dark side as these reviews and comments are vulnerable to manipulation. And as advances in AI continue, fake reviews generated by AI agents rather than users pose even more scalable and dangerous manipulation attacks. These attacks on online discourse can sway ratings of products, manipulate opinions and perceived support of key issues, and degrade our trust in online platforms. Previous efforts have mainly focused on highly visible anomaly behaviors captured by statistical modeling or clustering algorithms. While detection of such anomalous behaviors helps to improve the reliability of online interactions, it misses subtle and difficult-to-detect behaviors.

This research investigates two major research thrusts centered around manipulation strategies. In the first thrust, we study *crowd-based* manipulation strategies wherein crowds of paid workers organize to spread fake reviews. In the second thrust, we explore *AI-based* manipulation strategies, where crowd workers are replaced by scalable, and potentially undetectable generative models of fake reviews. In particular, one of the key aspects of this work is to address the research gap in previous efforts for anomaly detection where ground truth data is missing (and hence, evaluation can be challenging). In addition, this work studies the capabilities and impact of model-based attacks as the next generation of online threats. We propose inter-related methods for collecting evidence of these attacks, and create new countermeasures for defending against them. The performance of proposed methods are compared against other state-of-the-art approaches in the literature. We find that although crowd campaigns do not show obvious anomaly behavior, they can be detected given a careful formulation of their behaviors. And, although model-generated fake reviews may

appear on the surface to be legitimate, we find that they do not completely mimic the underlying distribution of human-written reviews, so we can leverage this signal to detect them.

Dedicated to my family – I love you.

ACKNOWLEDGMENTS

I want to give my biggest thanks to my advisor, James Caverlee, who suggested I work on this topic, and I finished only because of his supervision and leadership. He patiently walked me through every step of a research project in the early stage of my Ph.D. and taught me how to be an independent researcher and simultaneously develop mentorship skills. His role in my study, career and personal life's growth is indisputable. I am grateful for this opportunity forever.

My committee: Frank Shipman, Ruihong Huang, Patrick Burkart, thank you for your comments, feedback and suggestions that shaped my research.

The experience of doing a Ph.D. is memorable mainly because of the people I get to meet every day — I always keep everyone in InfoLab as my close friends and get back to them anytime in life. I want to start with my senior labmates Majid Alfifi, Wei Niu, Cheng Cao, Hancheng Ge, and Hoakai Lu. Thank you for welcoming me to the lab, sharing your experience with me, and continuing your support after graduation. Can't wait to reunite with you in Seattle and do hotpot together :) I also want to thank Jianling Wang, Ziwei Zhu, Yin Zhang, Yun He, Xing Zhao, and Zhuoer Wang. It was fun to work with you. In particular, I want to thank Majid Alfifi, for inspiring me to feel that nothing is impossible to achieve. My only regret from graduate school is not knowing Chinese, as a result I missed many of the research discussions going on in our lab :)

A big part of my Ph.D. time is my internships where, I was fortunate to intern at USC ISI, Snap Research, and Microsoft Research. Thank you, Fred Morstatter, Nichola Lubold, Neil Shah, Leonardo Neves, Maarten Bos, Saurabh Tiwary, Brandon Norick, and Gargi Ghosh. It was a pleasure to work with you all. In particular, I want to thank two of my greatest mentors, Neil Shah, for giving me this opportunity to work on user engagement in private networks from the ground up and for his advice during my job search and, Fred Morstatter, for inviting me to ISI for a talk and giving me this chance to work on cultural artifacts in organizations– the coolest project ever. As I am writing this part and reviewing the past, I can tell all these opportunities have a root in the beautiful city of Los Angeles in one way or another. So, thank you, LA, for being so kind to me :)

These internships are not complete without co-interns, who are now my best friends and we still keep the tradition of Snappy hours and talk occasionally. Thank you, Hemant Surale, Kamalakkannan Ravi, Brihi Joshi, Tanay Singhal, Meera Rachamallu, John Teng, Deniza Malataeva, and Aysha Cotterill, for being part of this journey. You are all fantastic. It won't be fair not to mention Biryani (an Indian food), the reason for working hard and staying late in the office :)

I want to thank my recommenders Murat Kantarcioglu, Anna Squicciarini, Jen Golbeck, Leman Akoglu, and Taeho Jung, for supporting my application for US permanent residency.

Also, I want to thank my master advisor Babak Sadeghiyan for supporting my application for grad school and Daniel Takabi for helping me to come to the US to pursue a PhD.

Foremost, I want to thank my family. Thank you, Mom and Dad, for your emotional and financial support. I would not be here without you. Likewise, my deepest gratitude goes to my husband for his endless love, support, and sacrifice. You were my biggest asset through thin and thick, the one I can share my concerns with and complain about everything without being worried:)

Last but not least, I want to thank myself for doing PhD, working hard, taking no days off and being persistent during the tough time :)

I can't finish writing this line without mentioning the pandemic and how it disrupted the last part of my study overnight! Hiring freeze, delay in green card process, delay in graduation, and the list goes on. Thanks, Covid, for teaching me the difficult lesson that it always can get worse and even targeting me in person after all :)

This acknowledgment is a bittersweet end to one of the significant chapters, and I want to finish it with a Persian poem: "There is much more to say, but there is no more space left in this book."

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professors James Caverlee, Frank Shipman and Ruihong Huang of the Department of Computer Science and Engineering and Professor Patrick Burkart of the Department of Communication.

All work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by a fund from Texas A&M University and NSF grants.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

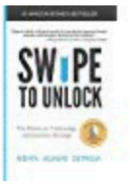
TABLE Page

1. INTRODUCTION

User generated content is ubiquitous. In particular, user review aggregators are a cornerstone of how we make decisions – from deciding what products to purchase (e.g., Amazon), movies to view (e.g., Netflix, HBO, Hulu), restaurants to patronize (e.g., Yelp), and hotels to book (e.g., TripAdvisor, Airbnb). Figure 1.1 shows an example of a review on Amazon along with other contextual information such as the reviewer, review title, time stamp and ratings. One study shows that 93% of consumers read reviews to determine the quality of a business [1]. This study also highlights the importance of user reviews through some key statistics: positive reviews influence 73% of users to trust a business, while 50% of users would question the quality of a business after reading negative reviews. In addition, policy makers rely on online commenting platforms like Regulations.gov and FCC.gov as a means for citizens to voice their opinions about public policy issues. Figure 1.2 shows an example of a comment posted to the FCC about net neutrality policy.

However, showcasing the opinions of fellow users has a dark side as these reviews and comments are vulnerable to manipulation. For example, about one out of five reviews on Yelp are suspected of being fake according to Yelp's proprietary filtering algorithm [2]. Furthermore, a recent study of comments posted to the FCC about repealing net neutrality found that over one million pro-repeal comments were likely faked; in contrast, a majority of the legitimate comments were in favor of keeping net neutrality [3]. And as advances in AI continue, fake reviews generated by AI agents rather than users pose even more scalable and dangerous manipulation attacks [4]. These attacks on online discourse can sway ratings of products, manipulate opinions and perceived support of key issues, and degrade our trust in online platforms [5].

With these challenges in mind, *this dissertation research investigates manipulation powered by humans and AI in large-scale review platforms and proposes frameworks to combat them.* Therefore, the dissertation is aligned along two major research thrusts centered around manipulation strategies. In the first thrust, we investigate *crowd-based manipulation campaigns*. In these campaigns, crowds of paid workers organize to spread fake reviews. In the second thrust, we investigate



★★★★★ from Parisa on March 31, 2019

Help you master storytelling

As a phd student in computer science, I read this book to refresh my mind about already known concepts. It was a great practice and I strongly recommend to anyone looking for a great source to either understand the technology basics or reinvigorate them.

Figure 1.1: Example book review on Amazon

Filing Detail

ID	109011305614746	**Proceeding**	17-108
Name of Filer	Orion Hunter		
Type of Filing	COMMENT	**Filing Status**	DISSEMINATED
Viewing Status	Unrestricted		
Date Received	Sep 1, 2020	**Date Posted**	Sep 1, 2020
Address	76 N main street	**City**	Marshall
State	NC	**ZIP**	28753
Brief Comment	It is imperative that Net Neutrality and title II remain intact as is, permanently.		

Figure 1.2: Example comment on FCC about net neutrality issue

AI-based manipulation campaigns, where crowd workers are replaced by scalable, and potentially undetectable generative models of fake reviews. In particular, one of the key aspects of this dissertation is to address the research gap in previous efforts for anomaly detection where ground truth data is missing (and hence, evaluation can be challenging). In addition, this dissertation investigates the capabilities and impact of model-based attacks as the next generation of online threats. For both crowd-based and AI-based manipulation, we study the capabilities and impact of these approaches, propose inter-related methods for collecting evidence of these attacks, modeling attack behaviors, and create new countermeasures for defending against them.

1.1 Research Questions and Contributions

Concretely, this dissertation research aims to address three overarching research questions:

- **RQ1** What are the characteristics of crowd-based manipulation attacks and how can we detect them? (Chapter 3)
- **RQ2** What are the characteristics of AI-based manipulation attacks and how can we detect them? (Chapter 4)
- **RQ3** How can we leverage AI for more sophisticated attacks? (Chapter 5)

In response to these questions, this dissertation defines six research objectives and makes the following specific contributions to achieve these objectives:

- **RO1:** Can we sample evidence of review manipulation and understand crowd manipulation behavior for development and evaluation of detection methods? This objective complements proposed methods in previous works which are typically evaluated based on i) manually labeled dataset, ii) simulation of fake review behavior, or iii) examination of highly visible fraudulent users.

  Contributions: In this dissertation, we implement a cross-domain paradigm to obtain a pool of crowd manipulation traces. In particular this framework: (i) monitors crowdsourcing websites that task crowd workers to write fake reviews on Amazon products, (ii) links these tasks to Amazon platform to identify the products that are targets of fake reviews, and (iii) crawls the reviews and reviewers associated to such products.

- **RO2:** Can we detect manipulators who seek to camouflage their behaviors? It is notable that crowd campaigns are small in size meaning they often target particular items with carefully targeted fake reviews, meaning they do not leave obvious traces of anomaly behaviors. Detecting manipulators would help system administrators to remove them and improve the quality of the reviews.

Contributions: In this dissertation, we propose a comprehensive framework called TwoFace to uncover crowd review manipulators. Two of the key components of TwoFace are: (i) propagating the "suspiciousness" of seed manipulators to identify nearby users through a random walk over a "suspiciousness" graph, and (ii) uncovering distant users who serve structurally similar roles by mapping users into a low-dimensional embedding space that captures community structure. TwoFace uncovers many manipulators even when the system is seeded with only a few fraudulent samples with 93% recall and 77% precision, outperforming the state-of-the-art baseline.

- **RO3:** Can we identify the targets of crowd review manipulation, rather than just individual review manipulators? Concretely, this research objective is to recognize manipulation patterns at the item level. Knowing which products (or services, etc.) are targets of an attack and understanding their review behavior, we can deploy more resources to defend the targets from ongoing threats (e.g., require additional user verification or enlist more human moderators). Therefore, we can develop more robust defensive countermeasures for future threats (e.g., by learning patterns of what is being targeted). In addition, understanding fake review behaviors helps us to develop robust recommendation techniques that diminish the impact of fake reviews on recommendation results.

  Contributions: We first propose a data expansion methodology to obtain a rich set of target products by leveraging the notion of suspicious *reviewers*. In the next step, we propose a comprehensive framework called TOmCAT abbreviated from **T**arget-**O**riented **C**rowd **AT**tack. TOmCAT is composed of three key components: (i) identifying crowd attack footprints and formulating them into a suite of features based on timing and ratings; (ii) embedding attack footprints in a 3-layer neural network to uncover target products; and (iii) finally, augmenting TOmCAT to counter strategic attacks that attempt to create hard-to-detect behavioral patterns by undermining timing-based footprints. TOmCAT uncovers target products with 84% accuracy and outperforms state of the art techniques.

- **RO4:** What are the characteristics of AI-powered review manipulation where crowd workers are replaced by generative models of fake reviews? And can we detect this type of manipulation? Especially, this class of attacks creates new challenges for defense mechanisms, including: (i) scalability, since AI approaches do not rely on paying workers, meaning large-scale attacks can be launched; (ii) increased deception, since AI approaches can potentially obfuscate signals left by crowd campaigns (e.g., by varying the rate of posting fake reviews) that are helpful for detection. However, generating high quality reviews that are readable by humans is a challenge on its own.

  Contributions: In this dissertation, we develop a comprehensive framework called DIOR comprising three main components. It first demonstrates the success of AI agents in generating high-quality synthetic reviews by leveraging neural language models. The main contribution here is to develop a universal model capable of generating user reviews across different domains benefiting from transfer learning techniques. In the next step, it shows that synthetic reviews are robust against linguistic-based spam detectors. More importantly, we conduct a user study that demonstrates these reviews are considered to be realistic by human judges. Finally, it proposes a discriminator leveraging distributed representation of the reviews to distinguish between synthetic and user written reviews. The major finding is that synthetic reviews successfully pass traditional computational detectors and human tests confirming the power of AI in review manipulation. However, the proposed defense mechanism is able to distinguish synthetic and real reviews with 90% accuracy.

- **RO5:** What are the differences between crowd and AI-based manipulation in terms of quality of reviews written by humans and reviews generated by models? This research objective seek to answer the important question of whether the proposed DIOR generator is capable enough to take the place of crowd campaigns.

  Contributions: We conduct a user study to evaluate how end users perceive the fake reviews generated by the proposed DIOR compared with those written by crowd manipulators. The

main finding is that users find model-generated reviews as reliable as fake reviews written by crowd campaigns.

- **RO6:** Finally, what is the potential of more sophisticated AI-based manipulation to go beyond the proposed DIOR framework? Addressing the fourth research objective, we face a new challenge that motivates us to develop new generative models in order to control the topic and sentiment of generated reviews. Although DIOR shows success in generating grammatically correct, meaningful and human-readable reviews, it does not reflect the desired aspect and sentiment of the reviews. It should be noted that content in review platforms are personalized for each item or service rather than the same content being duplicated across different items or services. Therefore, we propose to model a topic-aware generator while the main challenge is the lack of a review dataset with labels at topic and sentiment level.

  Contributions: In this dissertation, we develop a framework called ADORE (Aspect Dependent REview labeling) comprising two main components. It first proposes a weak labeling methodology to build a large-scale dataset of reviews labeled at aspect and sentiment levels. Then it proposes a joint neural model that encodes aspect and sentiment into the language model. We extensively evaluate the quality of assigned labels and generated reviews.

1.2 Structure of the Dissertation

This dissertation is structured as follows:

- Chapter 2 gives a comprehensive literature review and highlights the contributions of this dissertation in connection with prior work.

- Chapter 3 gives a thorough description of crowd-based manipulation and the curated dataset. It studies and evaluates the proposed frameworks to detect manipulators and targets of manipulation. In summary, research objectives 1 to 3 and their corresponding contributions are discussed in this chapter.

- Chapter 4 describes AI-based manipulation and gives the details of the proposed framework from generation, evaluation and mitigation aspects. It compares the capability of AI-based attacks against crowd manipulators. In summary, in this chapter we cover research objectives 4 and 5 and their corresponding contributions.

- Chapter 5 describes the components of the proposed framework for validating more sophisticated attacks. This chapter covers research objective 6 and its corresponding contributions.

- Chapter 6 summarizes the outcomes and findings in this dissertation and provides several open directions for future work.

1.3 Publications

Table 1.1 contains the list of main publications that are the result of this dissertation and have appeared in peer-reviewed top-tier web and data mining conferences. It also contains our proposal for future work.

Table 1.1: List of publications that are the result of this dissertation research

Publication	Chapter
[6] P. Kaghazgaran, J. Caverlee, M. Alfifi " Behavioral Analysis of Review Fraud: Linking Malicious Crowdsourcing to Amazon and Beyond", ICWSM'17.	3
[7] P. Kaghazgaran, A. Squicciarini, J. Caverlee "Combating Crowdsourced Review Manipulators: A Neighborhood-Based Approach", WSDM'18	3
[8] P. Kaghazgaran, M. Alfifi, J. Caverlee "TOmCAT: Target-Oriented Crowd Review ATtacks & Countermeasures", ICWSM'19	3
[9] P. Kaghazgaran, M. Alfifi, J. Caverlee "Wide-Ranging Review Manipulation Attacks: Model, Empirical Study, and Countermeasures", CIKM'19.	4
[10] P. Kaghazgaran, J. Wang, R. Huang, J. Caverlee "ADORE: Aspect Dependent Online Review Labeling for Review Generation", SIGIR'20.	5
[11] P. Kaghazgaran, J. Caverlee "Towards an Automated Writing Assistant for Online Reviews", CHI AutomationXP20 Workshop.	5
J. Caverlee, P. Kaghazgaran "Explainable Online Manipulation Detection", proposal submitted to Sony Inc.	6

2. RELATED WORK

In this chapter, we discuss related work from the aspects of traditional and model-based manipulation in line with the two major thrusts of this dissertation.

2.1 Traditional Manipulation

Propagation of false information has been studied on different platforms such as Twitter [12, 13, 14], Facebook [15, 16, 17] and Wikipedia [18]. This dissertation narrows the focus to deception in e-commerce and review platforms [19, 20, 21, 22, 23, 24, 25, 26, 27, 28].

Indeed, many previous efforts have explored methods to uncover review manipulation, often by applying techniques based on machine learning [19, 20, 21, 22], graph mining [23, 24, 25] algorithms, or via probabilistic modeling of user behaviors [26, 27, 28]. However, the research gap is that *the actual intent to deceive* is unknown and these methods typically are built and validated over a dataset of “known" manipulated reviews. In this section, we provide a taxonomy of the related work based on their evaluation methods and point out how this dissertation complements previous approaches and addresses this gap by understanding the intention behind crowdsourcing campaigns.

- *Manual labeling of fake reviews:* In the first approach, judges – often either researchers themselves or a team of labelers at a review site – assess individual reviews to determine if they are fake or not [29, 30]. These methods sometimes rely on unsupervised algorithms (e.g., the output of a proprietary company algorithm) or on manual and possibly error-prone labeling of fake reviews without access to a ground truth of the actual intent of the review writers themselves.
- *Ex post analysis of outliers:* A second approach is to validate detection algorithms through ex post analysis of suspicious reviews. Typically, an algorithm is run over a collection of reviews and the top-ranked results are examined [23, 24, 31, 32]. This approach tends to focus on highly-visible fake behaviors (e.g., a reviewer who posts dozens of reviews in a

period of minutes), but may miss more subtle behaviors.

- *Simulation of bad behavior:* The third approach is to *simulate* the behaviors of malicious workers [33] in which volunteers are asked to imagine themselves as fake review writers and then post fake reviews. This method lacks insight into the strategies and motivations of actual fake review writers. The simulation also can be done by injecting dense blocks into the matrix (or tensor) representing review data [34, 35]. However, these approaches may have difficulty in detecting subtle attacks where there are not such clearly defined dense block characteristics.

In contrast to these efforts where the critical knowledge of intent to deceive is missing, this dissertation research seeks to provide strong evidence of intent to deceive by building a ground-truth of actual review manipulators and targets of manipulation.

To detect review manipulation, typical approaches tend to adopt one of the following properties:

- *Text-based characteristics:* The *content of the reviews* may offer indicators of "spamness" or deception. Many works here develop NLP models to distinguish fake reviews from legitimate reviews [21, 33, 36, 29]. For example, linguistic features include, but are not limited to, similarity, stylistic, lexical, psycho-linguistic and sentiment characteristics [33, 19, 37, 20, 38, 39, 40].
- *Behavioral characteristics:* The reviewers or the reviews may leave clues as to which are fraudulent such as skewed rating distributions [41, 42] and dense inter-arrival times between successive reviews [41, 42, 27]. However, the methods to capture these clues focus on indirect approaches, where the ground truth is necessarily an approximation. In other words, they are only able to identify users with many reviews in a short burst of time or with skewed rating/inter-arrival time distributions as suspicious [43, 35, 28, 44] while many seemingly "normal" users may be overlooked in practice if their deceptive activities are blended in with legitimate ones. In contrast, this dissertation aims to identify users who have been tasked by a crowdsourcing site. In this way, we can identify with high confidence users who are indeed

deceptive even if the majority of their reviews are legitimate.

- *Network characteristics:* Graph-based approaches model users as nodes and their relationships as edges. Approaches include spectral methods like eigen-decomposition or singular value decomposition to cluster similar nodes in the graph [45, 46, 47]. Iterative models to label nodes as trustworthy or non-trustworthy are proposed in [23, 48]. Markov Random Fields (MRF) and belief propagation approaches have been used to identify dense and most likely suspicious sub-graphs [49, 24]. In another view, malicious users are detected through graph-based measures like neighborhood diversity [32]. A separate direction detects dense blocks in a ratings matrix [43, 34, 35], to find clusters of coordinating raters. Extraordinary dense blocks correspond to groups of users with lockstep behaviors, e.g., [43]. Moreover, this method has been lately extended from matrix to tensor representation to incorporate more dimensions (e.g., temporal aspects) [34, 35]. Contrary to dense block detection algorithms that focus on lockstep behavior – meaning a high number of fake ratings are required – a target of crowdsourcing manipulation (e.g., a product on Amazon) may be subject to dozens of fake reviews. Specifically, we find that the number of required fake reviews requested by paymasters varies: the average is 13, but some paymasters ask for only 1, while the maximum requester asked for 75 reviews and 84% of the tasks require fewer than 20 fake reviews. This indicates the challenge in identifying fraudulent behavior.

2.2 Model-based Manipulation

This section discusses related work with respect to model-based review manipulation from two angles of *generation* and *detection* and highlights this dissertation's contributions in connection with prior work:

Generation. Here, we first give an overview of the literature for automated text generation and then turn to review generation more specifically.

Natural language generation techniques place structured data into well-designed templates [50, 51]. These systems require rules and consistent format. Probabilistic approaches like N-gram models generate text by looking back only a few steps in the sequence [52]. While N-gram

models exhibit limitation against long text sequences, RNN-based models perform based on complex “memory" gating which maintain longer term dependency [53, 54, 55]. The application of RNN-based text generators on different domains like chatbot [56], conversational systems [57], email auto-responses [58], movie dialogues [59] and image captioning [60] has shown successful results. In another direction, Generative Adversarial Networks (GANs) have been explored to fill in the blanks in sentences [61]. They were originally designed to produce differentiable values that have direct application in computer vision, so generating discrete text is a challenge for them. On the other hand, researchers have shown that properly regularized RNN models can achieve state-of-the-art performance in generating text [62].

With respect to research on automated review generation, character-level RNNs to capture the sentiment and meaning in product reviews have been proposed in [63]. Byte-level RNN to generate product reviews for sentiment classification is proposed in [64]. An N-gram based review generator is examined in an adversarial setting with the purpose of generating fake reviews [65] without considering any countermeasures. In another effort, [4, 66] propose a character-level RNN to generate fake reviews for the specific domain of Yelp. [66] uses sequence-to-sequence neural models to incorporate contextual information such as location into the model.

In contrast to this dissertation, these previous works do not consider a universal model capable of generating domain independent reviews. The character-level model comes with its own issues like misspelling, requiring a large dataset, and days to converge which limits its applicability. In addition, such models cannot benefit from transfer learning and need to be built from scratch for the new domain as the characters remain the same across different domains.

Similar to our work, a few studies also explore the impact of transfer learning on NLP tasks. This includes work on sentiment classification [67], multilingual language modeling [68], machine translation [69] and question and answering [70]. To the best of our knowledge this is the first to explore the power of transfer learning in generating domain-independent online reviews.

Detection. Here, we first give an overview of the literature on detecting generated content in general and then discuss detecting automated review manipulation more specifically.

Statistical approaches like measuring TF-IDF or examining Zipf's law on the target text are proposed to detect machine-generated text with the focus on textual features [71, 72]. [66] uses N-gram features to classify reviews as real or synthetic. Recent advances in GANs that produce synthetic images suggest mimicking the underlying distribution of the image where the discriminator is responsible to distinguish between the distribution of true and generated samples [73]. In this dissertation, inspired by the idea of the discriminator in GANs, we propose to distinguish between real and synthetic reviews based on the distributed representation of their constituent tokens. It should be noted that the method proposed in [4] to defend against model-generated reviews examines the character distribution of synthetic and real reviews as the language model is trained at character level granularity. Therefore, the direct application of this approach on word-level generators is ruled out.

As mentioned earlier, the last research objective of this dissertation is to validate more sophisticated automated attacks on review platforms where it can control the topic and sentiment of the generated review. For this purpose, we provide a literature review over the techniques related to this problem and highlight the contributions offered by this dissertation. Hence, this related work is discussed from three dimensions: *aspect extraction*, *sequence to sequence modeling*, and *attribute-based review generation*.

Aspect Extraction: There is a large body of research to extract the aspects of products wherein users have expressed opinions in order to analyse the crowd sentiment towards these aspects [74, 75, 76, 77, 78]. These methods are based on relatively explicit representations of the text and attempt to model each topic as distribution of its words. Aspect extraction has been performed using methods like frequent pattern mining [74], topic modeling [75], word alignment [76], label propagation [77] and recently deep neural networks [78]. In contrast this dissertation aims to build a dataset of reviews at topic and sentiment levels for generating aspect-aware reviews and to do that it proposed to find the topic boundaries in a review so that it can segment a review into its aspect-specific parts

Sequence-to-Sequence Models. Sequence models are trained to convert an input sequence into a

target sequence. Applications like question answering [79], neural translation [80], chatbot [56], conversational systems [57], email auto-responses [58] and image captioning [60] have been developed under the umbrella of sequence-to-sequence architectures. However, due to lack of ground truth of aspect-aware reviews and non-sequential nature of the attributes, the direct application of seq2seq models in our problem domain is challenging.

Attribute-based Review Generation. In another direction, research has focused on generating product reviews from attributes [81, 82]. These models learn to generate customized reviews for each user based on history of their review writing. The input attributes are rating, product ID and user ID. This dissertation focuses on aspect as an attribute. There is a car review dataset [83] at the aspect level, where each review already contains eight sentences for eight aspects and the proposed model generates reviews that cover all the aspects. However, in general domains like restaurants and e-commerce platforms, users are not forced to describe all aspects of the target. Also, they may use several sentences to describe a single aspect which we refer to as segments. This dissertation aims to extract aspect-specific segments to build the ground truth.

3. CROWD-BASED MANIPULATION

In this chapter, we describe crowd-based manipulation from sampling manipulation traces to detecting manipulators and targets of manipulation attacks. In particular, we develop a cross-domain data collection approach to obtain samples of crowd manipulation and propose two comprehensive frameworks known as TwoFace and TOmCAT to detect manipulators and targets of manipulation respectively.

3.1 Sampling Review Manipulation Traces

The purpose of this step is to build the review manipulation ground truth at scale. The main insight is that evidence of "intent to deceive" is extremely difficult to determine from the review domain alone, and yet, critical to successfully uncovering manipulation. To address this challenge, we propose a cross-domain approach and develop two different crawlers to obtain evidence of manipulation and consequently collect associated reviews and reviewers' activities. Figure 3.1 demonstrates the workflow of crowd review manipulation wherein a worker visits the crowdsourcing sites, picks a task that asks users to write inauthentic reviews on review platforms, writes review and gets paid in return. The data collection is based on this workflow.

As an example, consider the task posted to RapidWorkers in Figure 3.2. This type of task is common on RapidWorkers and related sites like ShortTask and Microworkers. As an example of the type of review that is created by crowd worker, Figure 3.3 shows a sample of a crowdsourced review for a "cortisol supplement" product sold by Amazon. On examination, this review displays few (if any) clear signals of it being fraudulently written.

Hence, this dissertation focuses on tasks posted to RapidWorkers that target Amazon and simultaneously ask for fake reviews. Note that there are many such sites[1] and many additional targets (e.g., Yelp, App Store, Play Store).

Concretely, the first crawler monitors low moderation crowdsourcing sites, where pay-masters

[1] We also checked several other crowdsourcing websites such as ShortTask, Microworkers and Amazon Mechanical Turk (AMT); however, tasks related to promoting products in Amazon are mainly announced on RapidWorkers.

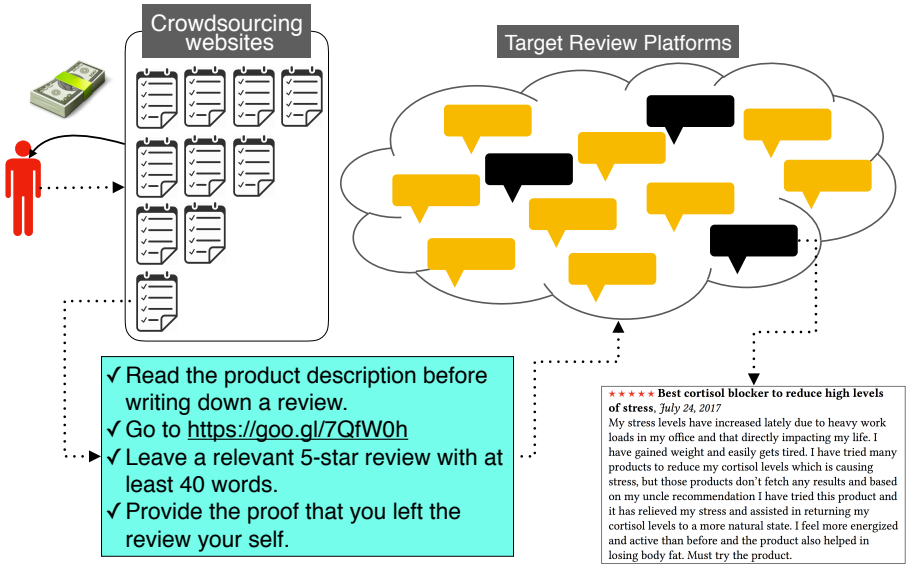


Figure 3.1: Crowd Review Manipulation Workflow. A worker visits the crowdsourcing sites, picks a task, writes a fake review and gets paid. Our data collection method is based on this workflow.

What is expected from workers?
Read the product description before writing down a review.
Go to https://goo.gl/7QfW0h.
Leave a relevant 5-star review with at least 40 words.
Provide proof that you left the review yourself.

Figure 3.2: An example crowdsourcing task.

★★★★★ **Best cortisol blocker to reduce high levels of stress**, *July 24, 2017*
My stress levels have increased lately due to heavy work loads in my office and that directly impacting my life. we have gained weight and easily gets tired. we have tried many products to reduce my cortisol levels which is causing stress, but those products don't fetch any results and based on my uncle recommendation we have tried this product and it has relieved my stress and assisted in returning my cortisol levels to a more natural state. we feel more energized and active than before and the product also helped in losing body fat. Must try the product.

Figure 3.3: An example review written by a crowd worker.

task crowd workers to write a few fake and highly positive reviews on a specific product on Amazon. The workers are paid a little (mostly less than $1) in return. Hence, it obtains a pool of crowd tasks which we know for certain have targeted review sites with fake reviews. By linking these tasks to the target review platform, the second crawler collects reviews associated with targets of manipulation and the activity history of their reviewers. In total, we were able to identify 900 tasks targeting 533 unique products. We call these initial products the *primary target products*. There are ∼33k associated reviews. Next, we explored the activity history of ∼14k reviewers and crawled their reviews, resulting in ∼580k reviews. Ultimately, our dataset contains the following information: product ID, review ID, reviewer ID, review title, review content, rating, and time-stamp.

To investigate the behavior of fraudulent vs non-fraudulent reviewers, we sampled reviewers with at least 10 reviews. For non-fraudulent reviewers, we sampled reviewers from the Amazon dataset introduced in [84], who co-reviewed products beyond those crowdsourcing products targeted by a crowdsourcing attack giving us ∼38k reviewers in total.

We further propose an expansion methodology to gain a richer set of target products using the notion of suspicious reviewers. Since not all reviewers on a target product are suspicious, we adapt a similar methodology to that introduced in [19] in which users with a certain fraction of duplicate or semi-duplicate reviews are labeled as spammers. To measure the similarity, we gauge the exact matching between two reviews. Therefore, we calculate the Jaccard function between each pair of reviews written by a single reviewer and take the maximum value as its *Self-Similarity Score*. Formally, the self-similarity score of reviewer v with review collection of R_v is defined as:

$$SSS(v) = max\{Jaccard(r_i, r_j) | \forall r_i, r_j \in R_v \wedge i \neq j\}$$

$$Jaccard(r_i, r_j) = \frac{|r_i \cap r_j|}{|r_i \cup r_j|} \times 100$$

We found 9,659 users with perfect self-similarity score and label them as suspicious reviewers. In the next step, we explore the products which received reviews from suspicious reviewers and pull out those with many reviews from suspicious reviewers and label such products as targets. Through

our expansion methodology, about 3,500 products are identified which we call *secondary target products*. Crawling reviews associated with secondary target products, a corpus of 2.6M reviews is collected. In the experiments, the threshold is set conservatively such that target products were targeted by many suspicious reviewers (which helps mitigate any errors in our original labeling of suspicious reviewers).

3.2 TwoFace Framework to Detect Manipulators

This dissertation proposes the TwoFace system to uncover crowdsourced review manipulators who target online review systems. Detecting manipulators would help system administrators to suspend or remove them and improve the quality of the reviews. In addition, detecting manipulators is a reliable proxy to identify the fake reviews written by them given this fact that crowd reviews demonstrate proper grammar and other evidence of being generated by actual people (not bots) and it would be challenging to rely on textual content to detect them.

The aim is to study the challenge of countering crowd review manipulation in order to detect the *actors* that enable such attacks. Concretely, the main contributions are:

1. We study genuine and fake review writers through statistical analysis of their behaviors. We observe that fraudulent users engage in a mix of legitimate and deceptive behaviors and do not show strong signals of deception. (Section 3.2.1)

2. Motivated by these observations, we propose to exploit *neighborhood-based* characteristics of the reviewers themselves. The intuition is that although individual behaviors may be easy to mask, the collective campaign organization may be uncovered by exploiting the network around reviewers and products. Based on this intuition, the proposed TwoFace approach propagates the suspiciousness of the original seed users to identify "similar" users through a random walk over a "suspiciousness" graph. In this way, we can identify users who may be likely to participate in deceptive review campaigns. (Section 3.2.3)

3. Given users who are distant in the "suspiciousness" graph will rarely be considered as deceptive reviewers, we propose an embedding-based approach to distinguish deceptive reviewers through their community embedding structure. Based on this intuition, the proposed TwoFace approach uncovers these (hidden) distant users who serve structurally similar roles by mapping users into a low-dimensional embedding space that captures community structure. (Section 3.2.4)

4. We evaluate the proposed framework from different angles and show that TwoFace uncovers many manipulators even when the system is seeded with only a few fraudulent samples with 93% recall and 77% precision, outperforming a state-of-the-art baseline. (Section 3.2.5)

3.2.1 Behavioral vs. Network Characteristics

Over the dataset described above, we first study genuine and fake review writers through statistical analysis of their behaviors. A suite of observations demonstrates that although malicious and benign reviewers occasionally behave differently, review manipulators engage in a deceptive mix of legitimate reviews (to build reputation and trust) and fake reviews (to cash in on this trust) as shown in Figure 3.4. In the following, we examine their behavior in terms of rating, review burstiness, review length and self-similarity.

Ratings. we begin with Figure 3.4a, which shows the ratings distribution for reviews written by the two types of reviewers. Echoing previous studies, e.g., [41], we see that crowdsourcing workers tend to write 4 or 5-star reviews. While crowdsourcing efforts could be targeted at suppressing the ratings for a competitor, we see instead that most efforts focus on promotion. Compared to legitimate reviewers, the rate of 5-star reviews is 20% higher for fraudulent reviewers.

Review Length. We see in Figure 3.4b the distribution of the review length in terms of number of words between the two groups. We can see that reviews by fraudulent reviewers are relatively short. Even though task requestors require a minimum number of words for payment, these graphs show that all reviews written by fraudulent reviewers are not necessarily a response to crowdsourcing tasks which often require a word count minimum.

Burstiness of Reviews. Intuitively, crowd workers may seek to complete several tasks in a short time to maximize their payoff. Hence, for each reviewer we measure the standard deviation of the timestamp for that person's reviews – we consider the last 10 reviews for reviewers with more than 10 reviews. we plot the distribution for this "burstiness" as seen in Figure 3.4c. In this case, a small standard deviation corresponds to many reviews being posted in a short time window, whereas a higher standard deviation corresponds to reviews posted over a long time period (and hence, lacking burstiness). Contrary to our hypothesis, burstiness of reviews is not a strong indicator to

distinguish fraudulent and non-fraudulent users.

Self-similarity. Finally, we measure how much a reviewer's language mimics previous reviews they have written. Perhaps fraudulent reviewers write according to a simple "template", and so new reviews tend to repeat language used in previous ones. Here, we measure the lexical overlap between each two sequential reviews (r_i, r_j) written by the same reviewer using the Jaccard similarity as $JS = \frac{|r_i \cap r_j|}{|r_i \cup r_j|}$.

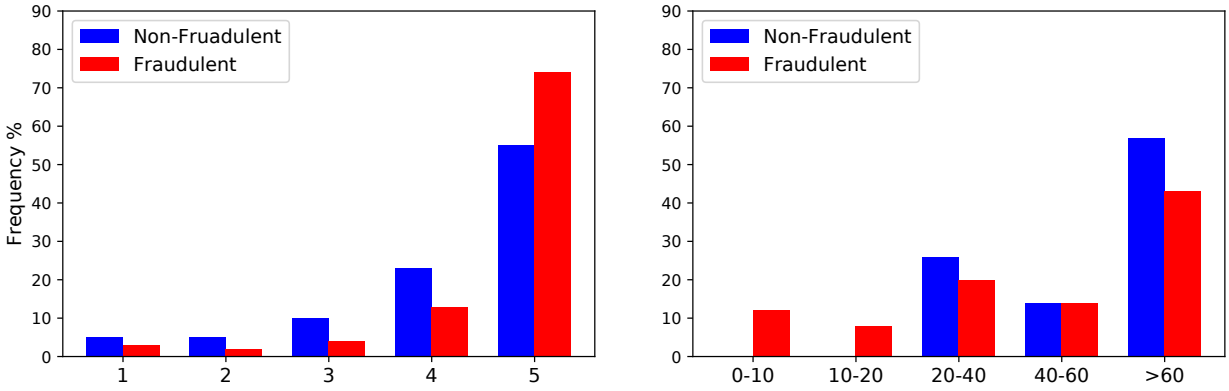
Figure 3.4d shows that non-fraudulent tend not to repeat themselves (low Jaccard score); whereas fraudulent reviewers tend to rely on repeated keywords or phrases. Intuitively, reviewers engaged in crowd-launched manipulation tend to mimic themselves over time since they are not actually experienced with the actual product.

Based on this mix of somewhat encouraging features, we evaluate a variety of classifiers (see Section 3.2.5). It turns out that these traditional features do a poor job of distinguishing these two-faced reviewers. The hypothesis is that traditional signals may fail since these reviewers engage in a mix of legitimate and deceptive reviews. Motivated by these observations, we turn to how we can propagate the suspiciousness of our original seeds for uncovering unknown fraudulent reviewers. We propose to exploit *neighborhood-based* characteristics of the reviewers themselves. The intuition is that although individual behaviors may be easy to mask, the collective campaign organization may be uncovered by exploiting the network around reviewers and products. In the following, we first describe the research hypothesis followed by the techniques for propagating suspiciousness and uncovering distant users.

3.2.2 TwoFace System Design

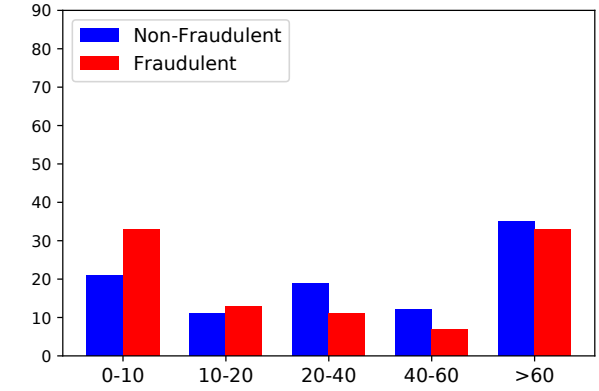
Figure 3.5 shows the overview of the TwoFace key components. It is designed to be deployed in real-world scenarios with the following characteristics:

- First, we assume there is only some *small, partial evidence of review manipulation*. In contrast to many previous efforts, we make no assumption that there is a large, curated collection of positive and negative examples of review manipulation. This corresponds to real-world scenarios of evolving and newly emerging types of review manipulation.

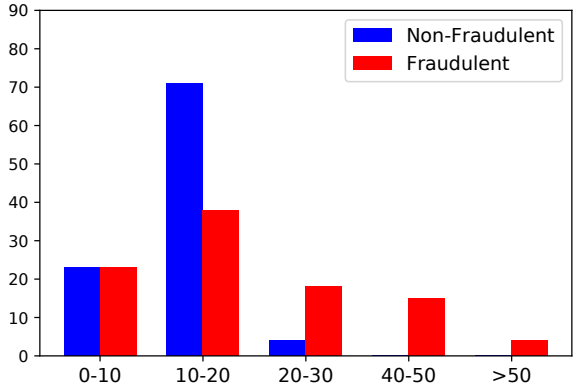


(a) Rating Distribution

(b) Review Length



(c) Review Burstiness in Days



(d) Self-similarity

Figure 3.4: Traditional features show some differences between fraudulent and non-fraudulent reviewers, but their distinguishing power is weak.

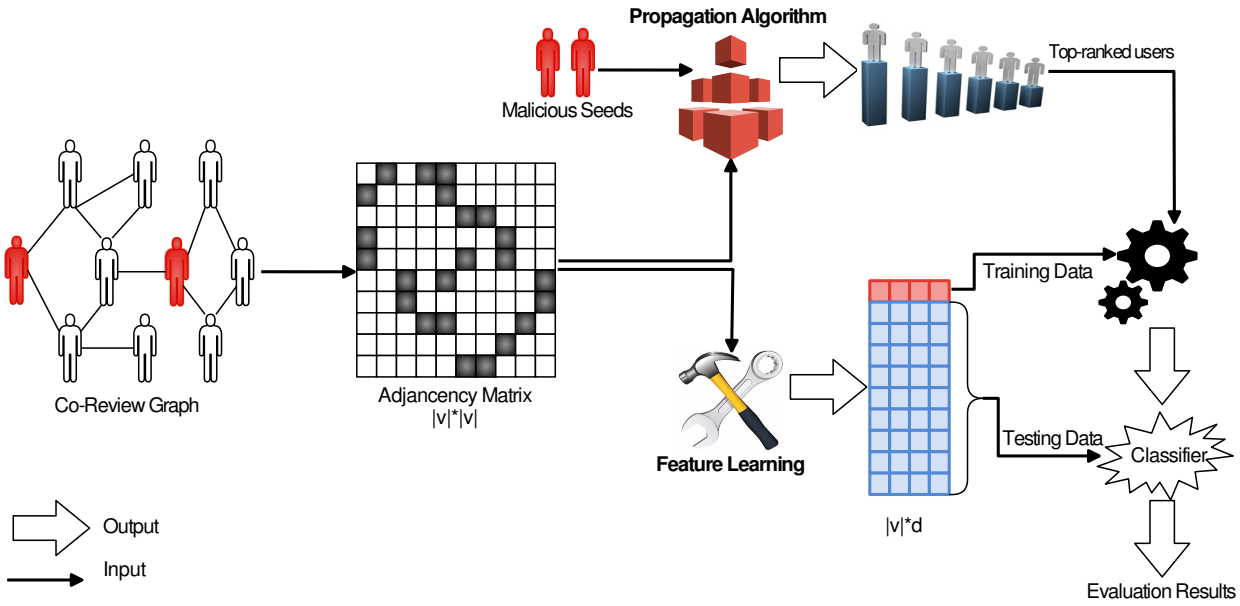


Figure 3.5: TwoFace overall framework.

- Second, we target scenarios in which review manipulation engages in *difficult-to-detect behaviors*. That is, the review manipulators may engage in a mix of legitimate and deceptive reviews, so that many traditional content-based or dense-block methods may have difficulty in finding them.
- Finally, TwoFace is *recall-focused*. The goal of uncovering review manipulators is to identify as many as possible out of the entire space of reviewers, in contrast to precision-focused approaches that may miss the vast majority of manipulators. we assume system operators can expend resources to further examine these potential manipulators.

3.2.3 Propagating Suspiciousness

In this and the following section, we propose two complementary perspectives on propagating the suspiciousness of a known suspicious user. The first – based on a traditional random walk over the user-user graph – exploits the locality of suspiciousness within the graph. The main intuition is that suspicious users will tend to cluster in the graph. The second – based on recent advances in network embeddings – exploits the structure of the graph around suspicious users. So even if two

users are distant in the graph, they may be considered similar in terms of their malicious activities. Intuitively, the campaign network structure around fraudulent reviewers in a site like Amazon may be similar even if the specific reviewers have not been encountered before.

Reviewer-Reviewer Graph. The approach is built on reviewer-reviewer interactions. These interactions are mapped to a co-review graph in which each reviewer is represented as a node and if two reviewers write a review on the same product, then there exists an edge between them. Formally, we model the reviewer network as a graph $G = (V, E)$ wherein V is a set of reviewers and E is a set of undirected edges that connect reviewers. In the experiments (see Section 3.2.5), we consider two scenarios: un-weighted and weighted edges. In the first setting, if two users u_i and u_j ($i \neq j$) have written reviews on multiple common products, this connection is represented as a single edge. In the second setting, we represent the number of common products as a weight value for the edge $w(u_i, u_j)$.

The number of nodes connected to user u is its degree $D(u)$. The co-review matrix corresponding to the un-weighted graph is calculated as:

$$m(u_i, u_j) = \begin{cases} 0 & \text{if } (u_i, u_j) \notin E \\ \frac{1}{D(u_i)} & \text{if} (u_i, u_j) \in E \end{cases}$$

Accordingly, the co-review matrix corresponding to weighted graph is calculated as:

$$m(u_i, u_j) = \begin{cases} 0 & \text{if } (u_i, u_j) \notin E \\ \frac{w(u_i, u_j)}{\sum_{k \in D(u_i)} w(u_i, u_k)} & \text{if} (u_i, u_j) \in E \end{cases}$$

Note that the values in each matrix are normalized transition probabilities.

Random Walk Propagation. The proposed approach for propagating suspiciousness is inspired by the $TrustRank$ algorithm proposed in [48]. we aim to compute a suspiciousness score for each user based on their connectivity to other users in the co-review graph. The intuition is that fraudulent users write reviews on similar products, so they may form a dense sub-graph of suspicious

reviewers.

The input to the algorithm is the co-review matrix (m), a seed set (s), teleportation parameter (α), the number of iterations (n), and the number of nodes ($|V|$). The output is a suspiciousness vector (r).

Algorithm 1 Suspiciousness Rank

1: $e = 0_{|V|}$ //Restart Vector
2: **for** $i = 1$ to $|V|$ **do**
3: **if** $u_i \in s$ **then**
4: $e(i) = 1$
5: $e = \frac{e}{|e|}$
6: $r = e$
7: **for** $j = 1$ to n **do**
8: $r = \alpha.m^T.r + (1 - \alpha).e$
9: **return** r

The restart vector e is initialized in steps 2-4 based on the seed set, i.e, the values are one in the corresponding indices of seed set items and zero in other places. Step 5 computes the $l1$ norm of vector e so that the aggregate sum of the vector is equal to 1. In step 6, the suspiciousness score vector r is initialized to e. Finally, the scores are calculated in steps 7-8 using a biased PageRank with e as a restart vector referring to the seed set of fraudulent reviewers.

In each iteration, the suspiciousness score of a node propagates among its neighbors and is dampened by the teleportation factor α. The score vector r shows the probability of reaching each node when the random walk, rooted at one of the nodes in the seed set, is traversing the graph. In other words, the final scores measure how relatively close a node is to the initial seeds. This approach ranks reviewers based on their level of suspiciousness and suggests reviewers that should be examined further by an expert.

Seed Selection. A key question is how the seeds for the random walk propagation are selected in the first place. First,we need to identify a small set of users as seeds that we certainly know

are fraudulent and then propagate their suspiciousness to other users using iterative approaches. In some works, e.g., [48], initial seeds are selected based on human judgment. Here, we consider three different scenarios for seed selection that could arise in practice:

- If we find a user with tens of reviews on targeted products and notice that all of his purchase statuses are unverified, then we conclude that this user is a critical player working on behalf of crowdsourcing websites. We call such reviewers *highly malicious users*. Therefore, we pick reviewers with the maximum number of such reviews as seeds. The intuition is that such reviewers may be connected directly to other potential fraudulent reviewers in the co-review graph. Therefore, propagating their suspiciousness would lead to identifying fraudulent reviewers more accurately. We call this approach the "best" choice of seeds.
- The second approach is to pick a few number of fraudulent reviewers randomly and propagate their suspiciousness. This approach is more indicative of real-world scenarios since we are not always guaranteed to have found the best (most connected) reviewers. For example, malicious users might use different accounts to write fake reviews in order to avoid detection models. We call this approach the "random" choice of seeds.
- The third approach is to pick a few number of fraudulent reviewers randomly among reviewers with only a few reviews on target products. The intuition here is that a system operator at a user review site like Amazon may have discovered some fraudulent reviewers, but only the weakest connected ones. How well does the suspiciousness propagation work in this case? We call this approach the "worst" choice of seeds.

In practice, we find that there is a great variance in the quality of seed selection approaches. To illustrate, Figure 3.6 shows the Precision@k over the top-k ranked reviewers when we initialize the suspiciousness propagation algorithm with seeds from the best, random, and worst cases. In the real-world cases of random and worst, we see that the fraction of reviewers that are actually fraudulent drops precipitously with an increase in k. That is, while there may be some localness in the co-reviewer graph which helps identify nearby fraudulent reviewers, many fraudulent reviewers are not closely connected to the seeds (hence, the low precision@k as k grows). We do see that

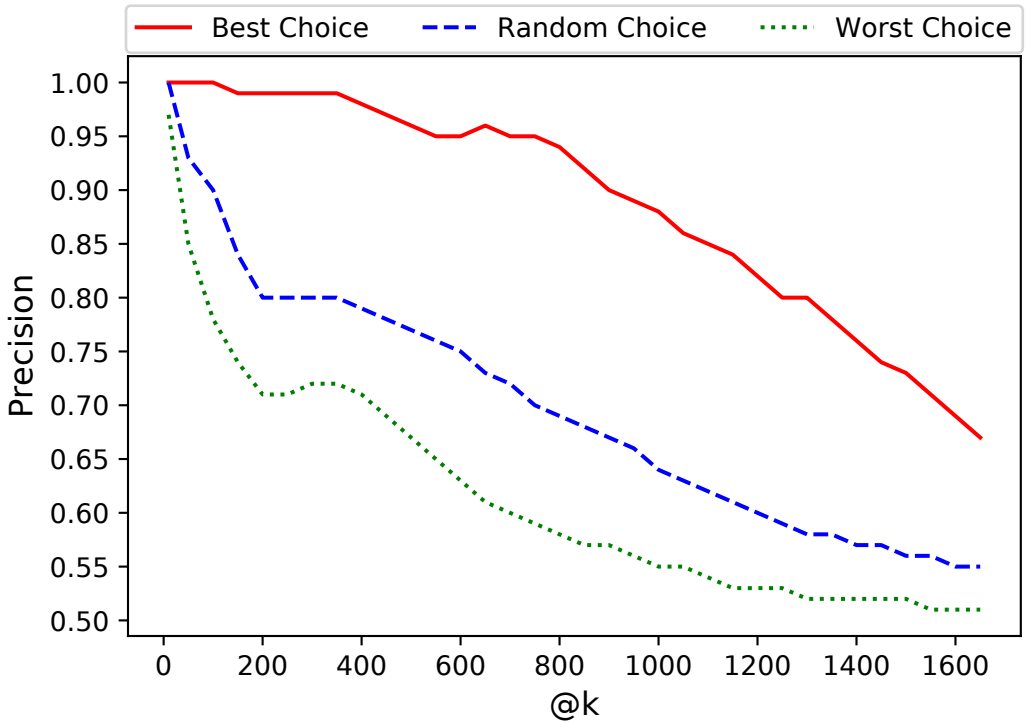


Figure 3.6: Precision@k for different seed selection approaches.

in the extreme case of picking the most prolific seeds (which we argue is rare in practice), there is good locality and good precision@k. Even so, for large choices of k, the quality drops off too.

3.2.4 Uncovering Distant Users

While traditional random walk methods exploit the locality of suspiciousness within the graph, they may miss reviewers who are not closely connected to the original seeds. In particular, in crowdsourcing scenarios, different fraudulent users may participate in different crowdsourcing tasks, so there would not be a direct link between them. Therefore, we need more sophisticated models to capture this relationship. In this section, we adapt a framework for learning feature representations of nodes in the network. Concretely, we explore using network embeddings to classify similar nodes even if two users are not directly connected but may have the same structural role in the graph. For example, Figure 3.7 shows that even though node 5 and node 10 act in two distinct communities, they play the same structural role in their own community.

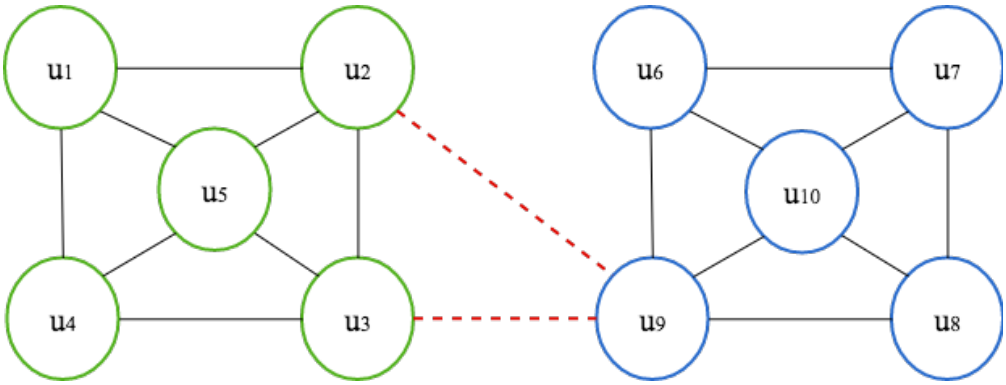


Figure 3.7: Same structural role in distinct communities. Here nodes 5 and 10 serve similar roles as we surmise different crowd campaigns may also be organized.

Reviewer Graph Embeddings. In node classification, the goal is to predict the most accurate labels for the nodes [85]. On the other hand, in supervised machine learning approaches, a set of informative features is required. When it comes to the problem of node classification in networks, it means feature representation of the nodes is required. The idea of network feature learning is inspired by the recent advances in natural language processing [86] such as the Skip-gram model. In summary, the Skip-gram algorithm goes over the words in a document, and builds a feature vector – the $word\ embedding$– for every word in the vocabulary such that it can predict its nearby words (i.e., words within a window). The continuous feature representation of words are learned by optimizing the likelihood objective function – Stochastic Gradient Descent (SGD) – and is based on the distributional hypothesis which declares words in similar contexts tend to be semantically similar [87]. In essence, similar words tend to have similar word neighborhoods.

Inspired by the Skip-gram model, a few recent works have proposed models for feature learning from networks where a network is "document" and the nodes are treated as "words" [88, 89, 90]. Similar to a document that is an ordered sequence of words, a network can be turned into an ordered sequence of nodes.

We adapt the recent learning approach proposed in [90] in which feature learning in networks is formulated as a maximum likelihood optimization problem. Function $f :\rightarrow R^d$ maps nodes to their feature representation which are used in classification tasks later on. d indicates the number of

dimensions of the feature vector, so f is a matrix of size $|V| \times d$. For each node $u \in V$, $N(u) \subset V$ is the set of its neighborhoods. Depending on the neighborhood sampling strategy, $N(u)$ includes either immediate, embedding, or mixture neighbors. The goal is to optimize the objective function given by f:

$$max_f \sum_{u \in V} log\, p_r(N(u)|f(u))$$

This function maximizes the log-probability of observing $N(u)$ as the neighborhood of node u given its feature representation $f(u)$. In our case, each node is a reviewer. Assuming the probability of observing n_i in neighborhood of u is independent from observing any other node in the neighborhood of u, the probability function can be treated as:

$$pr(N(u)|f(u)) = \prod_{n_i \in N(u)} pr(n_i|f(u))$$

Moreover, node u and its neighbors n_i have equal effect over each other in the feature space. Therefore, their conditional probability is modeled as a *softmax* function with dot product of their feature vectors:

$$pr(n_i|f(u)) = \frac{exp(f(u).f(n_i))}{\sum_{v \in V} exp(f(u).f(v))}$$

Putting it altogether, the objective function in Equation 3.2.4 can be re-written as follows wherein $Z_u = \sum_{v \in V} exp(f(u).f(v))$:

$$max_f \sum_{u \in V} \Big\{ - log\, Z_u + \sum_{n_i \in N(u)} f(u).f(n_i) \Big\}$$

The remainder is to wisely determine neighborhood nodes i.e., $N(u)$. The notion of neighborhood in a text document is defined by a sliding window over sequential words. However, due to the nature of networks, they do not have such a linear representation and a new notion of neighborhood is needed. Grover et. al in [90] proposed an efficient sampling strategy known as $node2vec$.

Traditional neighborhood sampling approaches are Breadth-first Sampling (BFS) and Depth-first Sampling (DFS). BFS samples nodes which are in immediate neighborhood of node u, while DFS samples ones which are in increasing distance from u. $node2vec$ enables interpolating between BFS and DFS. Briefly, a biased random walk explores the neighborhood of a node in a mixture of BFS and DFS ways by tuning two parameters – Return and In-out – parameters. In our case, we transform each reviewer (node) into an embedding that captures its neighborhood structure.

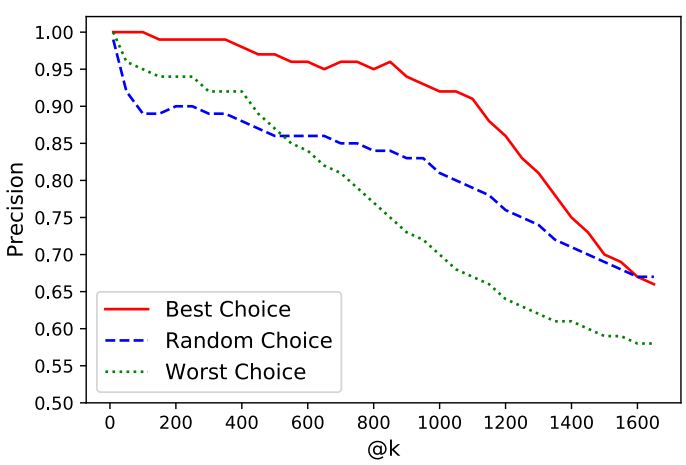
Putting it all Together. So far, we have suggested two complementary approaches to address the problem of uncovering fraudulent reviewers. The first one is based on traditional ranking algorithms which scores users based on their suspiciousness and is completely unsupervised. The other is an embedding feature representation of the reviewers. Here, we propose to combine these two approaches into a supervised classification framework:

- First, we take as input our seeds sampled from a crowdsourcing platform. Typically, we may have one to dozens of seeds.
- Then we propagate the suspiciousness of these reviewers via the random walk.
- After transforming every reviewer into its graph embedding representation, we then train a classifier where the positive examples (the fraudulent reviewers) are drawn from the ranked list of suspicious users. The negative examples (the legitimate reviewers) are drawn randomly from a held out set of reviewers.

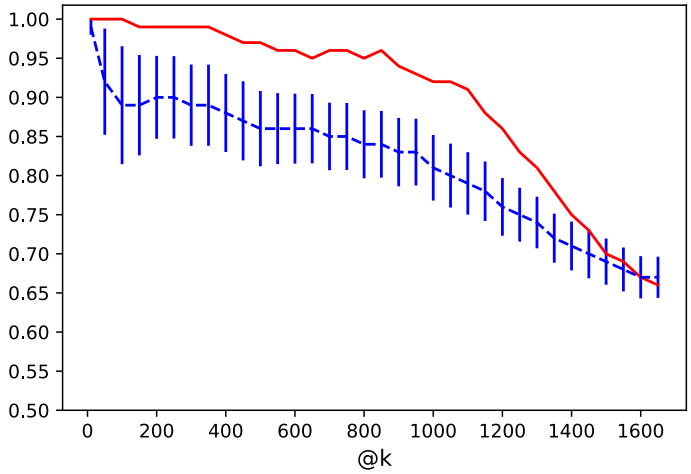
3.2.5 Experiments and Discussions

In this section, we first introduce the data labeling approach. Then, we report the results of evaluating the complementary methods to identify active users in review manipulation tasks. Finally, we compare our approach with a number of alternatives.

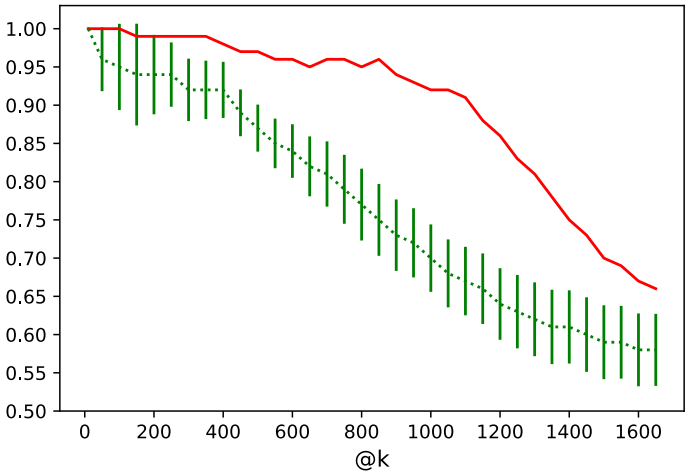
Ground Truth. In total, Table 3.1 shows the number of reviewers who wrote a review on a specific number of target products in our dataset (which include 12,212 reviewers in total). For example, 87 reviewers wrote reviews on more than 20 products targeted by the crowdsourcing site. Our dataset naturally contains a mix of reviewers and their reviews: some are legitimate reviews, some



(a) All three seed selection approaches.



(b) Random selection with error bars.



(c) Worst selection with error bars.

Figure 3.8: Precision@k for un-weighted graph with different seed selection approaches and 5 initial seeds.

are the result of targeted crowdsourced efforts, while others may also be fraudulent but outside the purview of our sampling method (e.g., launched via an unobservable channel like private email). Hence, we create two datasets – one based on a conservative assumption, and one that is a slight relaxation:

# Targets	1	2	3-5	6-8	9-20	>20
# Reviewers	9,096	1,669	1,093	126	141	87

Table 3.1: Distribution of reviewers based on number of target products they are associated with.

Conservative definition of fraudulent reviewers. we consider a reviewer to be a *fraudulent* if they have reviewed **two or more products** that have been targeted by a crowdsourcing effort. Intuitively, workers may aim to maximize their income by participating in many tasks (and hence, targeting many products). On the other hand, it is unlikely that a random user will write a legitimate review on two different crowdsourcing products in a short period of time, considering Amazon's selection of millions of products [91]. Making this conservative assumption, in our sampled dataset, we identify 1,650 of 38,590 reviewers as *fraudulent* and label the rest (36,940 reviewers) as *non-fraudulent*. Note that, 4,565 reviewers labeled as non-fraudulent still wrote one review on a target product. Of course, there may still be some unknown fraudulent reviewers in this set of 4,565 reviewers, but it gives us a baseline to compare against the clearly prolific fraudulent reviewers.

Relaxed definition of fraudulent reviewers. In an alternative way to identify fraudulent users, we can relax our conservative assumption and instead label all the reviewers associated with crowdsourcing products (i.e., 6,215 of 38,590) as *fraudulent* and label the rest as *non-fraudulent*. In this way, any reviewer who has reviewed a targeted product is labeled as fraudulent. While certainly overstating the number of fraudulent reviewers, this relaxed definition may give us more insights into the capability of our approach.

3.2.5.1 Propagating suspiciousness

Revisiting the initial approach to identify fraudulent reviewers by propagating suspiciousness, we report here an additional experiment where we consider five seed users. Again, we identify the best, random, and worst choice of seeds. we repeat this selection 20 times (since users are randomly chosen either from the entire set or from the least connected reviewers) and report in Figure 3.8 the variation of precision@k for different approaches in picking initial seeds. Here, we report all three approaches in (a), then we show the random approach with error bars in (b), and the worst approach with error bars in (c). The variability suggests that seed selection alone cannot identify fraudulent reviewers. This echoes the previous figure (see Figure 3.6), yet here we see that the "worst" approach can sometimes do better than random as we increase k. We attribute this result to the fact that some of the fraudulent reviewers have been more active in crowdsourcing manipulation in the past and so they are uncovered as actual fraudulent reviewers at $k <= 500$.

3.2.5.2 TwoFace detection

Given these results, we now turn to evaluating the quality of the end-to-end TwoFace system. Recall that TwoFace takes as input the seeds, the suspiciousness propagated scores, and the graph embeddings for the co-review graph.

Choice of Classifier. The first question is what classification algorithm to apply for distinguishing between fraudulent and non-fraudulent reviewers? Here, we consider six alternatives: logistic regression, SVM, Naive Bayes, Decision Tree, Random Forest, and a one-class version of SVM. One-class SVM builds a classifier using instances from only one class – fraudulent in our scenario – and then classifies new instances from both classes. we include this alternative to validate our use of legitimate reviewers in the training, even though our sample of legitimate reviewers is taken from a random sample of Amazon (and so, may erroneously include some fraudulent reviewers). Figure 3.9 shows the performance of different classifiers in the presence of 10% labeled data, where the results are averaged over 20 runs. we report the Precision and Recall for the Fraudulent class as well as F1-macro for the whole dataset. It should be noted that the other class which is abundant

(95% of the dataset) always has high Precision and Recall. We observe that Logistic Regression and one-class SVM give us higher Recall, 91% and 88% respectively, but Logistic Regression performs better in Precision. While Random Forest and SVM give higher Precision, 84% and 73% respectively, they have a poor Recall i.e., many of fraudulent users remain unidentified. Therefore, we do the further analysis using Logistic Regression.

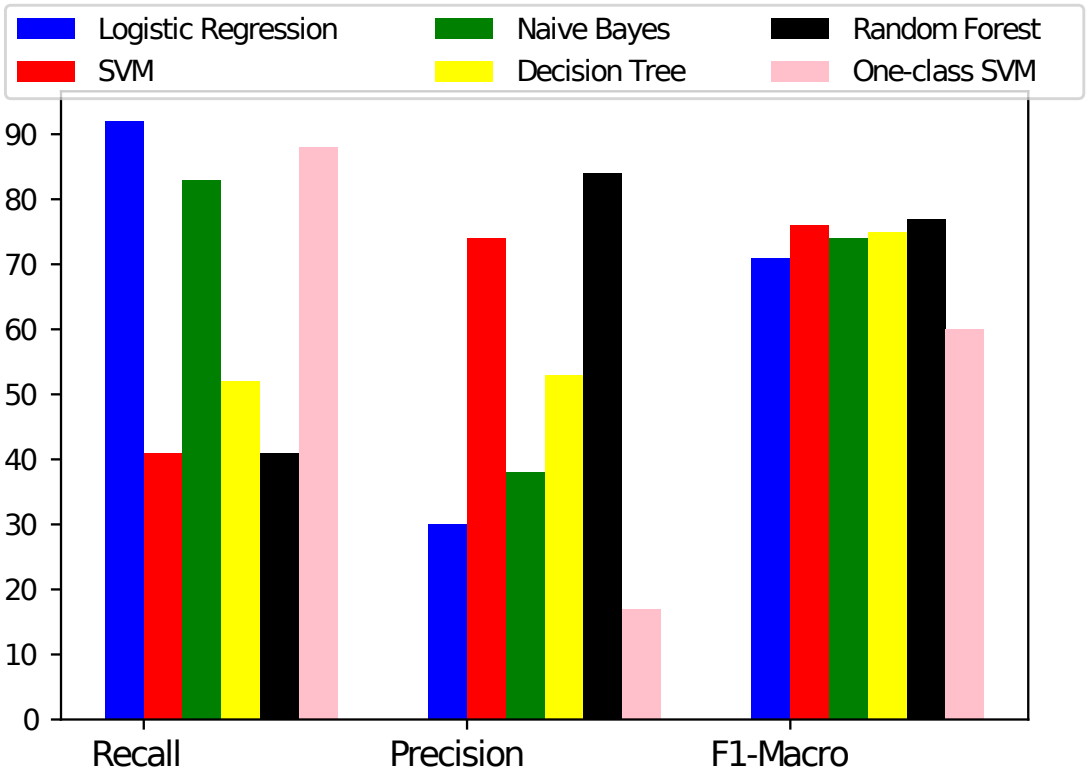


Figure 3.9: Comparing classifiers.

How Many Seeds Do We Need? In practice, the TwoFace system may have access to only a small fraction of all fraudulent reviewers. Hence, we study here the impact of the number of fraudulent reviewers on the quality of classification. Specifically, we consider samples of our entire dataset ranging from 1% to 10% of all reviewers; in this case, 1% corresponds to about 16 fraudulent reviewers and 370 non-fraudulent ones. we additionally consider both the weighted and unweighted graph for suspiciousness propagation as input to the method. Table 3.2 shows their

performance in the presence of 1% to 10% of labeled data. As we increase the number of seeds, Recall increases from 76% to 92% and 83% to 93% in un-weighted and weighted cases respectively while Precision remains relatively constant – 30% to 35%. This encouraging result shows that TwoFace can obtain high Recall even in the presence of very few seeds. Since the weighted and un-weighted cases perform relatively close, we focus the discussion on the un-weighted graph going forward.

Training	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
un-Weighted Graph										
Recall	76	85	88	90	91	91	92	92	92	92
Precision	34	33	32	30	30	30	30	30	30	30
F1-macro	72	71	70	70	70	70	70	70	70	70
Weighted Graph										
Recall	83	88	89	89	92	92	92	93	93	93
Precision	35	33	34	34	32	32	32	32	32	32
F1-macro	73	72	72	72	71	71	72	71	71	72

Table 3.2: Increasing the number of seed users available to TwoFace.

The Impact of Suspiciousness Propagation. In an earlier experiment, we saw that suspiciousness propagation alone is insufficient for uncovering fraudulent reviewers. Here, we investigate the impact of using the highly-ranked reviewers that are output from the suspiciousness ranking approach as input to our TwoFace classification. That is, do we really need to propagate suspiciousness? Or can we just use the initial seeds we sample from the crowdsourcing platform alone? In Figure 3.10, we show the impact of suspiciousness propagation on feature embedding classification. For example, if we rely only on the original seeds from the crowdsourcing platform we achieve around 71% for Recall of fraudulent reviewers; in contrast, if we feed a small number of highly-ranked users from the suspiciousness propagation algorithm into classifier, we see that Recall jumps to 83% with just 1% of seeds. This finding demonstrates the importance of propagating the suspiciousness of these seed users through the random walk over the co-review graph.

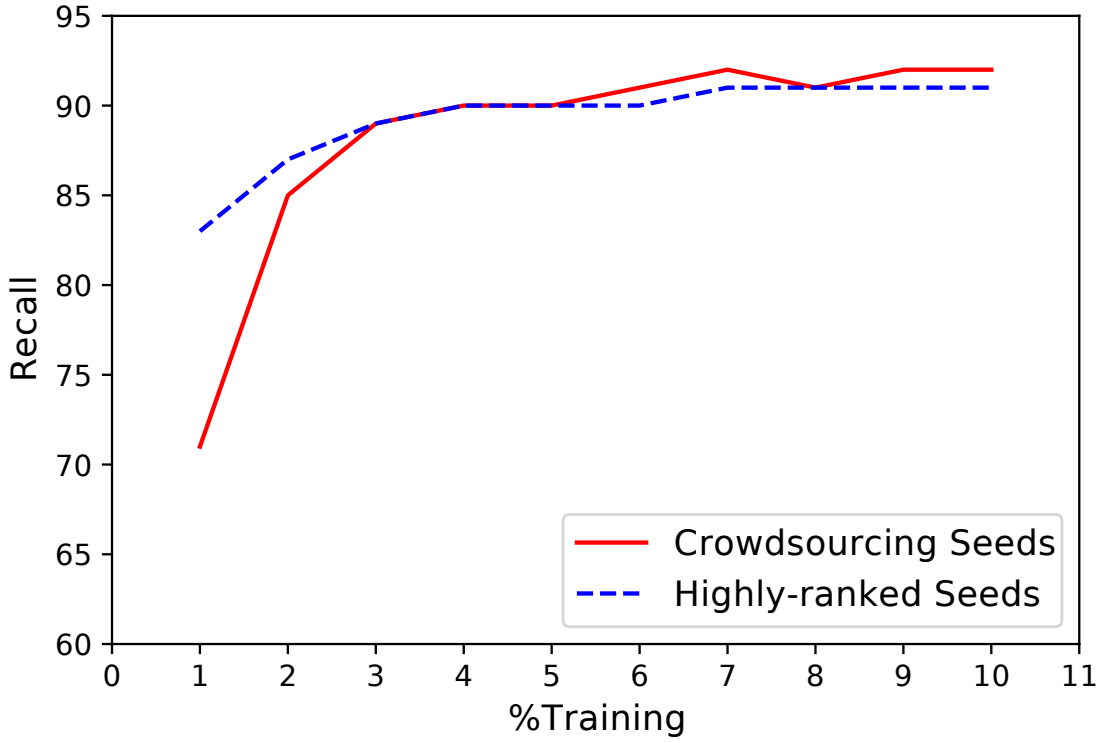


Figure 3.10: Impact of suspiciousness propagation in identifying fraudulent users

Relaxing the Ground Truth. As we can see from Table 3.2 Precision is in the 30% range which may seem surprising (even though Recall is quite high). This means that many of the non-fraudulent users have been misclassified as fraudulent, which could burden resources that are required to further examine these reviewers. To further explore this issue, we evaluate the TwoFace approach on the alternative "relaxed" ground truth in which every reviewer with at least one review on a target product is considered as fraudulent. In this alternative scenario, we find in Table 3.3 that precision jumps to 77%, while Recall increases even more to around 90%. This encouraging result suggests that many of the reviewers who only reviewed one target product historically are connected to other fraudulent reviewers and may need further examination by an expert.

Comparing with Alternatives. Finally, we compare the proposed TwoFace system with alternative methods including classification over traditional features and the state-of-the-art D-cube method [35]. For traditional features, we take the standard features described in the previous

	Recall	Precision	F1-macro
Conservative	83	35	72
Relaxed	91	77	89

Table 3.3: Relaxing the ground truth definition.

analysis of fraudulent reviewers, including rating, burstiness of reviews, review length, and self-similarity. we evaluate a variety of classifiers and report the best results which are from Logistic Regression. For D-cube – a dense block detection approach – we try many different number of dense blocks and report its best results when the number of blocks is 30 and 40. For a fair comparison, we also consider the relaxed ground truth for these methods. For TwoFace, we adopt Logistic Regression and use the highly-ranked users from the suspiciousness propagation as seeds.

	Recall	Precision	F1-macro
TwoFace System	**91**	**77**	**89**
Traditional Features	61	24	54
D-cube [35]/ 30 blks	69	34	50
D-cube [35]/ 40 blks	82	24	64

Table 3.4: Comparison of TwoFace with alternatives.

Table 3.4 shows that TwoFace system outperforms the two other approaches. The low performance of traditional features – 61% Recall and 24% Precision – indicates that fraudulent reviewers do not always behave abnormally and their rating distribution or burstiness might be similar to non-fraudulent reviewers. On the other hand the D-cube approach [35] which aims to detect dense blocks, i.e., a group of reviewers who write a review on a specific number of products in a short time, is the most similar scenario to crowdsourcing manipulation. It takes number of blocks as input and returns the most dense blocks. As a result, by increasing the number of blocks it returns more fraudulent reviewers – 82% versus 69% Recall with 40 and 30 blocks respectively. However, by increasing the number of blocks, we see that many non-fraudulent reviewers will be misclassified as fraudulent – 24% and 34% Precision. Since crowdsourcing campaigns do not

form dense blocks, we see that TwoFace provides the best overall performance with 91% Recall and 77% Precision.

3.2.6 Conclusion

The TwoFace system explored how monitoring tasks on sites like RapidWorkers can uncover fraudulent reviewers on sites like Amazon. It complements previous efforts by providing a new approach for identifying these types of reviewers. The main intuition is to: (i) exploit the locality of suspiciousness within the graph through a random walk to find suspicious users who tend to cluster; and (ii) exploit the structure of the graph around suspicious users to uncover campaign network structures for identifying fraudulent reviewers who are distant in the graph. The results are encouraging, indicating that TwoFace can indeed uncover many fraudulent reviewers.

3.3 TOmCAT Framework to Detect Targets of Manipulation

While fake review writers and fake reviews serve as a building block of an attack, the ultimate goal is often to manipulate a specific *target*. For example, the target of an attack could be a product (e.g., an item on Amazon), place (e.g., a restaurant on Yelp), service (e.g., web hosting service), or issue (e.g., a net neutrality post on FCC.gov). Knowing which products (or services, etc.) are targets of an attack, we can deploy more resources to defend the target from ongoing threats (e.g., require additional user verification or enlist more human moderators) and develop more robust defensive countermeasures for future threats (e.g., by learning patterns of the types of issues targeted). Furthermore, reviewers may have little or no history, say by using multiple user accounts to write fake reviews, degrading the impact of network characteristics on uncovering manipulation. More importantly, understanding fake review behaviors helps us to develop robust recommendation techniques that diminish the impact of fake reviews on recommendation results.

And yet, it has traditionally been challenging to identify which products (or places, services, issues) are actually targets of attacks without a gold standard dataset. As explained in Section 3.1, this dissertation builds a ground truth of *Amazon products* that have been targeted by crowd review manipulation attacks.

The main aim is to study the challenge of countering *target-oriented crowd attacks* in order to detect the target of attacks, in a complementary direction to approaches that focus on building blocks of attacks like fake reviews or fake review writers. Concretely, the main contributions are:

1. We identify two target-oriented attack patterns known as: (i) *promotion attacks*, wherein a crowd seeks to manipulate the product rating of a new product; and (ii) *restoration attacks*, wherein a crowd seeks to counteract a low rating from a legitimate reviewer. (Section 3.3.1)

2. With these attacks in mind, we develop a **T**arget-**O**riented **C**rowd **AT**tack detection framework called TOmCAT comprising two components: (i) it first formulates crowd attack footprints into a suite of features based only on timing and sequence of product ratings; (ii) it then embeds attack footprints in a 3-layer neural network, where we find a good success in

uncovering target products on our original Amazon dataset. (Section 3.3.2)

3. We show that TOmCAT outperforms six unsupervised and supervised baselines that originally were proposed to detect manipulation at the user/reviewer level. we find that review manipulation behaves differently at user and product levels. (Section 3.3.3.1)

4. We validate TOmCAT over three additional domains – Yelp, the App Store, and an alternative Amazon dataset – where we find that TOmCAT can effectively detect manipulation patterns in other domains. (Section 3.3.3.2)

5. Although TOmCAT can uncover target products with high accuracy by addressing existing attacks, strategic attackers can potentially create hard-to-detect behavioral patterns by undermining timing-based footprints. Inspired by recent advances in recurrent neural networks, we further propose a complementary approach to TOmCAT called TOmCATSeq.[2] This is the first exploration of RNN models on rating sequences for review manipulation detection. (Section 3.3.4)

3.3.1 Target-Oriented Crowd Attacks

We say that a coordinated attack that aims to manipulate a specific target is a *target-oriented crowd attack*. Such coordinated attacks have historically been difficult to identify in order to build a solid ground truth and evaluate corresponding countermeasures. Table 3.5 reports the summary of the dataset this dissertation curated and is explained in Section 3.1.

To model deception at the item level, a thorough understanding of crowd review manipulation behavior is required. we observe that these attacks demonstrate the following characteristics which help to formulate their patterns precisely.

Relatively small campaign size. A group of workers who target a specific product form a crowd campaign. A majority of the campaigns are small, soliciting between 5 to 10 reviews in total. This suggests that crowd campaigns do not leave obvious patterns of synchronized behavior that could aid in their detection.

[2]The naming is inspired by sequencing nature of RNN models

Table 3.5: Summary of Our Dataset

	# Products	# Reviews
Primary Targets	533	33 k
Secondary Targets	3,467	2.6 M
Randomly Sampled	4,000	0.7 M

⋆ *DON'T DO IT*, **February 15, 2018**
This is an inferior product. They are miniature, at least half the size they should be. we don't know anyone with eyes this small....
⋆⋆⋆⋆⋆ *Very Very good Product*, **February 16, 2018**
Very Nice product. Love it. Looking very cute. Best deal at best price. Very must satisfied with product and service.
⋆⋆⋆⋆⋆ *They really last for a good ..*, **February 17, 2018**
These lashes are so easy to use and they really last for a good while. There's a learning curve involved to apply them, ...
⋆⋆⋆⋆⋆ *Eyelashes look very natural*, **February 17, 2018**
I bought it for my girlfriend and she loves it. its a little difficult to learn how is the right way to put it on, but when you got .....
⋆⋆⋆⋆⋆ *Looks extremely real!*, **February 17, 2018**
I bought this for my girlfriend and we told her to test whether i could tell if they are her real eyelashes for fake one, but .....

Figure 3.11: Restoration Attack Example

High-quality reviews. Our fake reviews demonstrate proper grammar and other evidence of being generated by actual people (and not bots), meaning that existing methods that rely on signals of poor review quality may be ill-suited to uncover such fake reviews.
Therefore, we focus on crowdsourcing scenarios as a subtle attack where attackers do not leave obvious manipulation footprints and their human-written reviews make them difficult to detect.

Crowd Attack Types: In proposed dataset, we identify two prominent types of crowd attacks:

Restoration Attacks: If a product receives a low-rate review (1 or 2 stars in a 5-star rating system) it might be targeted by a crowd attack with highly positive reviews to help restore the overall rating. Figure 3.11 shows an example of this attack wherein the target product is first rated low by a 1-star review and then receives a series of 5-star reviews.

Promotion Attacks: In other cases, crowd attacks target newborn products, i.e. immediately

⋆⋆⋆⋆⋆ *best shower head we have gotten*, **August 30, 2017**
This multi-functional head helps us relax ourselves during the bath and gives the feel of taking a spa. we am using it everyday...
⋆⋆⋆⋆⋆ *Best product...*, **August 30, 2017**
The design of the product is amazing and we absolutely love the water flow combinations.The water flow settings can be...
⋆⋆⋆⋆⋆ *very high quality product*, **August 30, 2017**
The water flow settings can be changed easily and a very high quality product.It has a setting for a very full vigorous...
⋆⋆⋆⋆⋆ *Multifunction shower.....All in one*, **August 31, 2017**
Multifunction shower is very useful because Three Settings Water Flow Control For Your Pleasure. In this product ...
⋆ *Don't waste your Money*, **March 10, 2018**
Very cheap quality. After a few days, it became loose and detached from the hose. The plumbing tape even didn't help..

Figure 3.12: Promotion Attack Example

after the product is first introduced to Amazon. These early fake reviews aim to promote the product and encourage actual consumers to make a purchase. Figure 3.12 shows an example of such a scenario wherein the review thread is initiated by several fake reviews. However, it receives a low star review a few months later presumably from a true customer.

Initial Observations: Given crowd attacks characteristics and types, we analyze the rating behavior of target products versus randomly selected products from three dimensions. Products are described by their time-series ratings. More formally, for each product p, there is an ordered sequence of ratings as $R_p = (r_1, .., r_n)$ where r_i happens earlier than r_{i+1}.

1. Dense Review Behavior: First, we investigate how many reviews may turn up during a specific window of time w under the two classes of products. By sliding w over a sequence of reviews, we measure the number of reviews that are written in this interval. Referring to Figure 3.11 which gives an example of restoration attack, fake reviews were written within a time window of 3 days. Therefore, we set the value of w to be 3, 5 and 7 days. Also, since we observe a campaign size n is in the range of 5 to 10 reviews, we examine what portion of products receive 5 to 10 reviews within w days. Figure 3.13 summarizes the review behavior across different values of w and n. Interestingly, a large portion of target products demonstrate dense review behavior. For example, 78% of target products have received 7 reviews within 5 days while only 40% of

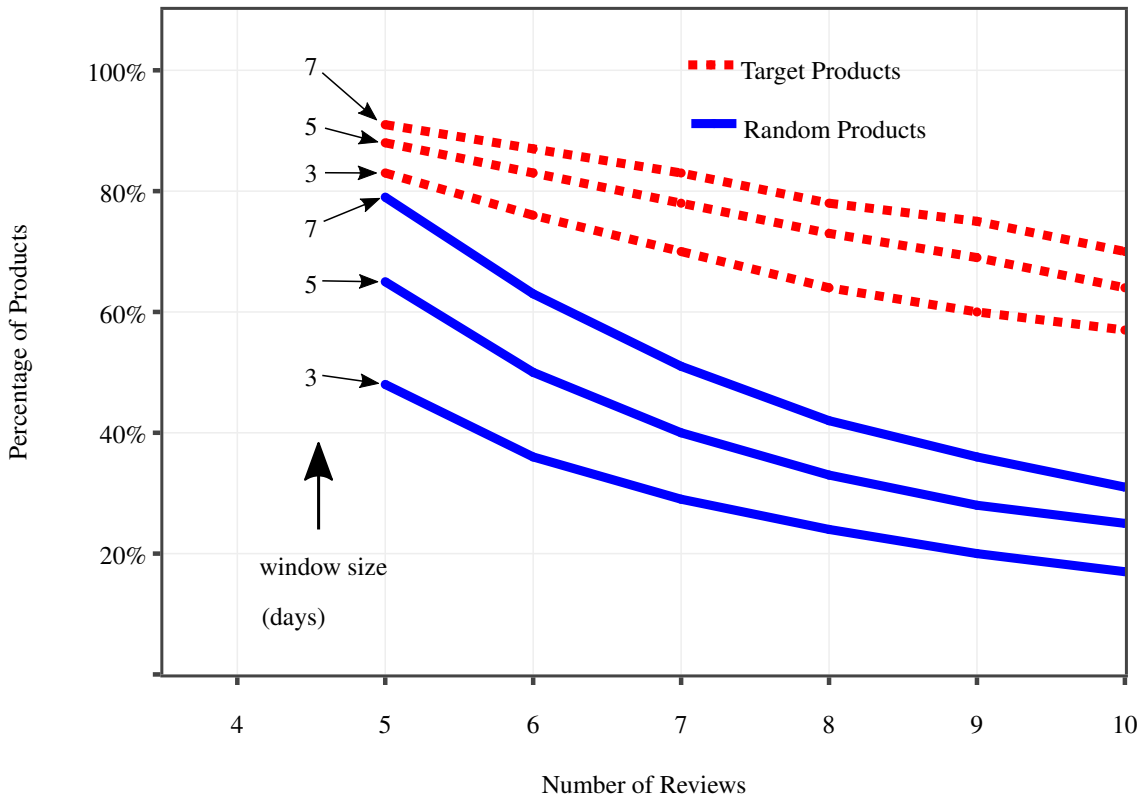


Figure 3.13: Target products tend to receive higher number of reviews in short time intervals.

random products display such behavior.

2. *Low High Rate Behavior:* The purpose of restoration attacks is to rebuild the trust in a product that has been shaken by a negative review. Thus, we investigate low-high rate events, e.g., a 5-star review showing up immediately after a 1/2-star review. In total, we found $111,870$ and $29,205$ number of such events in target and random products respectively. That is, target products are almost four times more vulnerable to this event. However, the existence of this kind of event is not a strong indicator of anomalous behavior as it could happen naturally due to consumers with different tastes evaluating a single product significantly differently [92].

To control for this, we measure how fast different products react to a negative review by gauging the inter-arrival time between sequential low and high ratings. Figure 3.14 shows what portion of

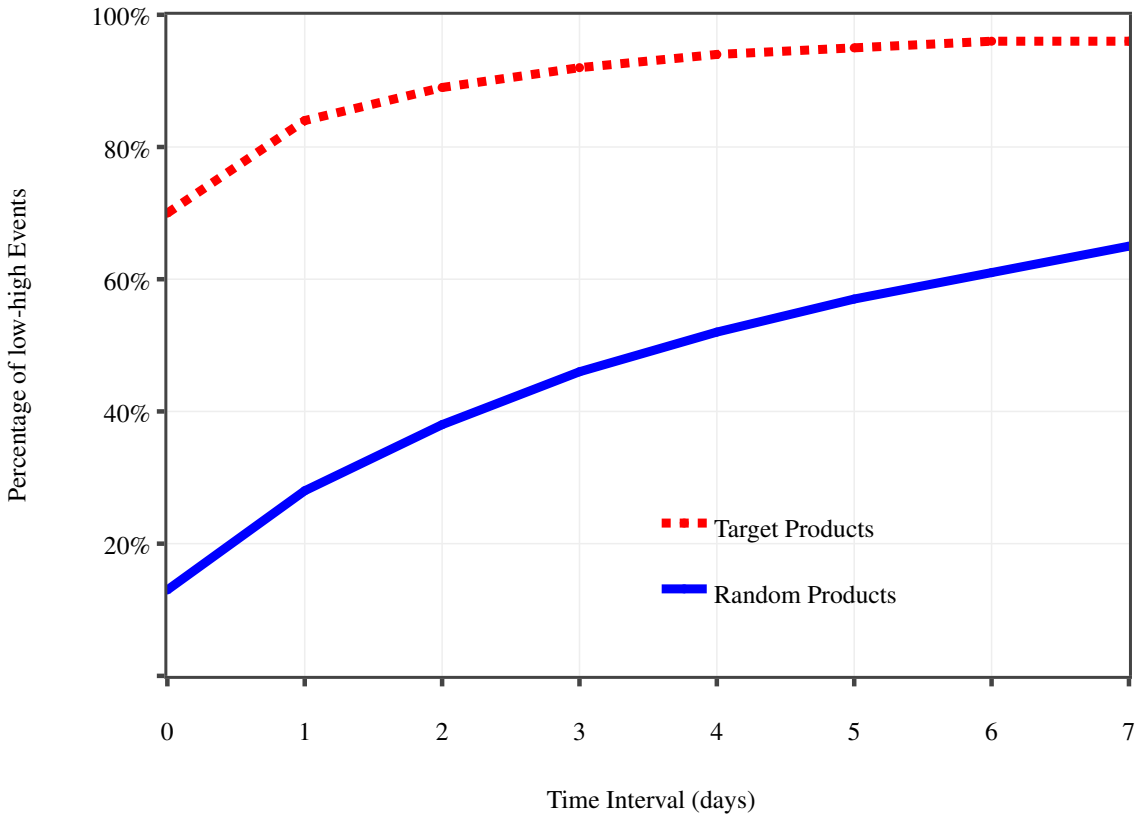


Figure 3.14: Target products react to low-rate review faster

these events happens in less than a specific time window w. we set w to be between 0 and 7 days. Interestingly, 71% of events occur in the same day as the negative review is written in target products while in random products, only 13% of such events happen in the same day.

Rough Estimation of Attack Types: 3,258 products experience low-high rate events. Considering same day occurrence of such event as baseline, we can say 2,660 out of ,3467 target products are targeted by restoration attacks. Similarly, 211 of the target products do not experience low-high rate events where we can say they are only targeted by promotion attacks.

3. Sequential High Ratings Behavior: This analysis counts the number of 5-star reviews immediately following a low-rated review. we set the size of these blocks of 5-star reviews to be between 5 and 10 with respect to the crowd campaign size. We can see from Figure 3.15 that the existence of such blocks in target products is about 5 times more likely than among their randomly selected peers.

3.3.2 Proposed TOmCAT Model

Inspired by these findings, we propose in this section the **T**arget-**O**riented **C**rowd **AT**tack detection framework (TOmCAT for short). The key intuition is to model targets based only on the timing and sequencing of product ratings, without access to historical reviewer behavior, reviews, and network properties. we introduce here the overall framework and a series of crowd attack footprints.

TOmCAT Structure. It is based on a neural network with three fully-connected layers. The input layer is fed attack footprints (as we describe next). The output layer has a single unit with labels 0 or 1 for each class of products (target vs. non-target). we use $ReLU(x) = max(x, 0)$ as the activation function in the hidden layers, a common choice in the literature [93]. The activation function for the output layer is a Sigmoid function which represents the classification result. Further, we adopt standard L2 regularization and gradient descent optimization.

We formulate two types of crowd footprints: *Micro features*, wherein rating behavior is modeled for a given product; and *Macro features*, wherein the deviation of rating behavior from a

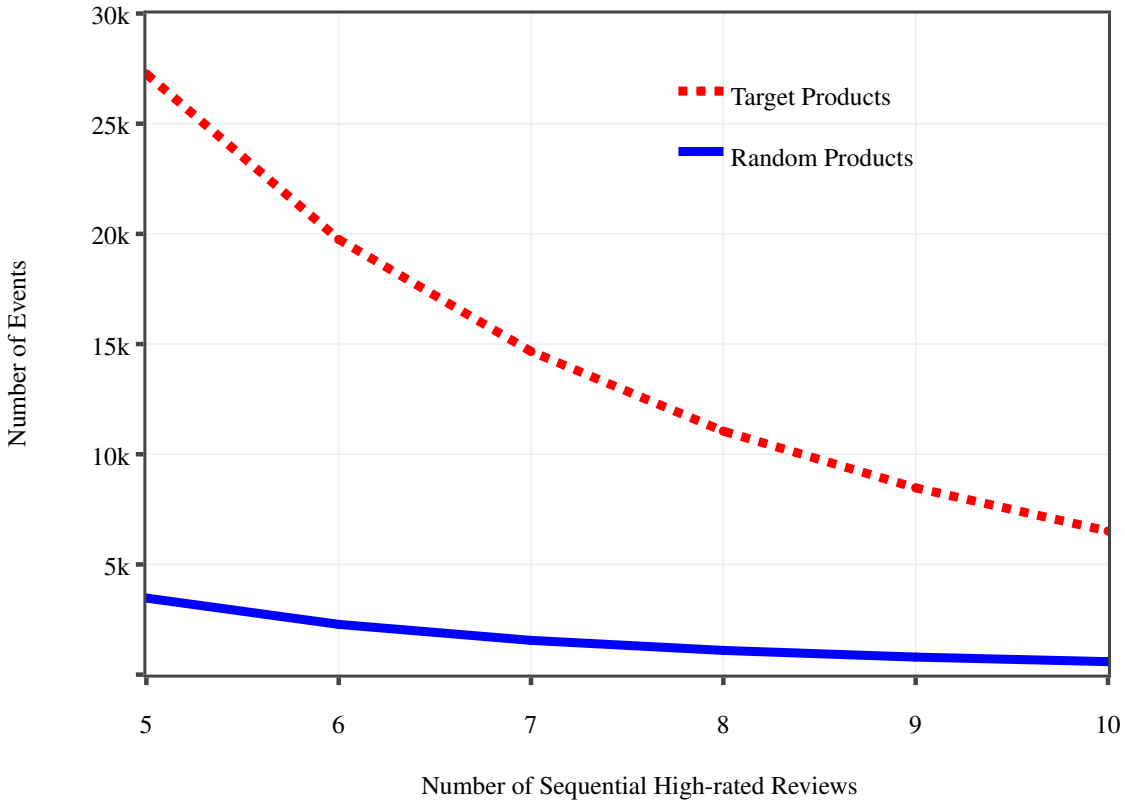


Figure 3.15: Target products tend to have more series of high-rating reviews immediately following a low-rating review.

base-model is measured.

3.3.2.1 Micro Features

Micro features codify rating patterns on individual products as follows:

Speed of Low-High Rate Events (SLH): In restoration attacks, the assumption is that target products receive high rate reviews faster to facilitate their rating recovery in the aftermath of a negative review. This feature is modeled as the average of inter-arrival times (IATs) between sequential low and high rate reviews.

$$SLH(p) = avg(IAT(r_i, r_{i+1})|r_i \in \{1, 2\} \wedge r_{i+1} \in \{5\})$$

Sequential High Ratings (SHR): Next we aim to capture how many 5-star reviews turn up after a negative review. This feature is modeled as the average number of sequential 5-star reviews following a low star review. The intuition is that while in target products the number of such reviews is limited to crowd campaign size, this feature does not carry any constraints in randomly sampled products.

$$SHR(p) = avg(k|r_i \in \{1, 2\} \wedge r_{i+1}, .., r_{i+k} \in \{5\})$$

Ratio of High Rating Reviews (RHR): As fake reviews are generated rapidly while in a normal situation reviews arrive more randomly, this feature measures the ratio by dividing SHR feature by its duration.

$$RHR(p) = \frac{SHR(p)}{IAT(r_i, r_{i+k})}$$

Variance of Inter-arrival Times (VIT): This last micro feature measures how inter-arrival time varies among ratings associated to a specific product. The intuition is that target products at some points of their life receive fake reviews rapidly and then reach an equilibrium state in which they

no longer exhibit abnormal behavior. we model this behavior by taking the standard deviation between the median and maximum values of inter-arrival times for each product. This feature has relatively greater value in target products.

$$VIT(p) = STD(median(IATs), max(IATs))$$

3.3.2.2 Macro Features

The macro features consider ratings in the context of a neighborhood of related products to measure the deviation from a base-model.

Base-model: we use an Amazon dataset [84] including ~83M reviews associated with ~1.5M products spanning from May 1996 to July 2014 to build the base-model. For example, the average rating distribution over all products in this dataset could be treated as base rating behavior. However, different products do not follow similar distributions, e.g., due to different quality, so relying on a single baseline does not provide a fair comparison. Therefore, we apply k-means clustering on base distributions to cluster similar samples together and scatter distant ones in different clusters.

Measuring Deviation: we use Kullback-Leiber (KL) divergence to compute the relative entropy between two probability mass distributions (PMD) similar to proposed approaches in [42, 94]. Formally, the KL-divergence between base-model (M) and a distribution (P) of a given product attribute is defined as:

$$KL(P, M) = \sum_{i=1}^{n} P_i \times \log_2 \frac{P_i}{M_i} \tag{3.1}$$

For example, if M and P are the probability mass distributions of ratings, then M_i and P_i indicate the probability of rating i in the corresponding distributions where $1 \leq we \leq 5$. The 0 value for KL-divergence means two distributions are identical while larger values indicate higher discrepancy between them. We refer to the value of KL-divergence as a product's anomaly score. Adding the notion of clustering, a direct approach for calculating an anomaly score would be to

aggregate KL-divergence between P and each cluster as follows:

$$\sigma(p) = \sum_{k=1}^{K} \rho_k \times KL(P, M_k) \tag{3.2}$$

where K, M_k and ρ_k indicate the number of clusters, probability mass distribution of the center of the cluster k and proportion of samples in cluster k, respectively. However, the output of the function is dominated by large clusters. For example, if a new sample is closer to a cluster with small size and far from large clusters, then the anomaly score becomes large even it is similar to one of the base behaviors. To address this issue, we modify the distance function to reinforce the impact of distance rather than cluster size inspired by *Inverse Distance Weighting* methods.

$$\sigma(p) = \begin{cases} 0, & \text{if } \exists k, KL(P, M_k) = 0 \\ \dfrac{1}{\sum_{k=1}^{K} \frac{\rho_k}{KL(P, M_k)}}, & \text{otherwise} \end{cases} \tag{3.3}$$

We introduce three attribute distributions where their deviations from the base-model form our macro features.

Rating Distribution: Since crowd review manipulation jobs solicit highly positive reviews, we assume that the rating distribution for target products should be skewed to high ratings. Considering 5-star rating system, rating distribution P_r is the probability mass function $[P_{r_1}, P_{r_2}, P_{r_3}, P_{r_4}, P_{r_5}]$ where P_{r_i} is the probability of observing i-star rating calculated as:

$$P_{r_i} = \frac{|r_i|}{\sum_{j=1}^{5} |r_j|}$$

Since crowd reviews often appear in a row, investigating inter-dependency behavior provides significant information about the existence of manipulation. Inspired by this property, the remaining macro features model inter-dependency features.

Inter-arrival time Distribution: Since crowd reviews often turn up in short period of time, their

inter-arrival times deviate from the base-model. Similar to time-stamp, inter-arrival time is not categorical data, requiring further care. We adapt the approach proposed in [41] to discretize the value space of continuous attributes. The intuition is that if maximum value of IATs (5 years in our case) is larger than minimum value (0 day) with an order of magnitude, then the value space is split logarithmically into d buckets:

$$d = \log_{base} max(IAT)$$

In the experiments, we set the value of the logarithm base to be 2, and as a result the number of buckets is 15.

Inter-arrival rating Distribution: Since a low-rate reviews can trigger restoration attacks and crowd reviews are highly rated, there should be a significant difference between two sequential ratings. We model this property as inter-arrival rating (IAR). The intuition is that the rating gap in target products reviews is higher than random products. On a 5-star rating system, possible values for IAR are $[-4, -3, -2, -1, 0, 1, 2, 3, 4]$ and thus, the probability mass function contains 9 discrete values.

3.3.3 Experiments and Discussions

This section evaluates the impact of different crowd footprints on the TOmCAT approach. Do macro features work well? Or micro features? We also explore the impact of features based on the first few reviews (which may be significant for detecting promotion attacks) and temporal-based features.

Experimental setup: The feed-forward neural network parameters such as hidden layers' dimension, regularization parameter λ and learning rate α are chosen via parameter tuning and we report the best results. Hidden layers are set to be 5 and 7 dimensions. λ and α are set to be 0.8 and 0.02 respectively. I apply normalization on input data to facilitate the convergence process before feeding them into the neural network. we train the network over 30% of samples and then evaluate its performance over the remaining 70% samples, where the results are averaged over 20 runs.

Table 3.6: Performance Evaluation of TOmCAT using Feedforward Neural Network (NN) in Different Settings.

	Target Products			Random Products			
	Recall	**Precision**	**F1**	**Recall**	**Precision**	**F1**	**Acc**
Macro Features	83	81	82	81	83	82	**82**
Micro Features	69	83	75	86	74	79	**77**
Macro + Micro	**86**	**83**	**84**	**82**	**85**	**84**	**84**
w/o Early Ratings	81	82	81	82	81	82	**81**
w/o Temporal	70	69	70	69	70	70	**69**

Results: Table 3.6 reports the results of TOmCAT for different types of crowd footprints. we report Precision, Recall and F1 score for each class of products and overall accuracy. we consider all of the micro features, all of the macro features, and both macro and micro features. As we expected, the aggregation of micro and macro features performs better in identifying target products with 86% recall and 84% accuracy. In contrast, TOmCAT identifies target products with 83% and 69% recall in the presence of only macro features and only micro features respectively. It is evident that a comprehensive detection framework boosts performance.

Further, we consider a special case where we drop all features based on the first few reviews of all products. Extracting features from the complete series of reviews can successfully model restoration attacks but it can miss promotion attacks, specifically ones targeting new products. By definition, these scenarios create circumstances that are not yet optimal for the model to detect. Therefore, we execute the proposed feature extraction methodology only on the first n reviews in addition to the complete series of reviews. In the experiments, we set the value of n to be 5 to cover the majority of this type of attack considering crowd campaign size typically varies from 5 to 10 fake reviews. We see in Table 3.6, fourth row, that performance metrics drop in the absence of early ratings features confirming the real impact of addressing this scenario. For example, recall in recognizing target products decreases from 86% to 81%.

3.3.3.1 Comparison with Baselines

In this section, we compare TOmCAT to six existing baselines which are originally designed to identify spam-behavior at the user level and we are eager to evaluate their performance at the product (target) level. These works aim to identify fraudulent users using rating and temporal features. To evaluate unsupervised approaches, we use precision $@k$ by varying k from 100 to 3,000. To adapt our model with precision $@k$, we examine the anomaly scores obtained from Sigmoid activation function at the output layer of the feedforward neural network and then sort the products in descending order.

- *Helpfulness Vote* [31]: On Amazon users can provide feedback to the reviews via helpfulness votes. We assume that target products receive fewer helpfulness votes. This approach uses the average of helpfulness votes of reviews of each product and ranks them in ascending order based on their helpfulness score.
- *BIRDNEST* [41] models temporal gaps and rating deviations.
- *edgeCentric* [42] also models temporal gaps and rating deviations.

Figure 3.16 illustrates that TOmCAT is superior to its alternatives especially as k increases acknowledging its capability in identifying most target products. For example, precision varies from 98% to 87% for different values of k. These results indicate that careful feature modeling can be important for defending against crowd attacks.

We further compare TOmCAT with the following supervised approaches.

- *SpamBehavior* [95] uses the average of rating deviation of individual ratings from overall rating as a feature.
- *Spamicity* [31] takes review-burstiness and maximum reviews per day as features.
- *ICWSM'13* [29] describes each user (or product in our setting) as its fraction of positive reviews, maximum reviews per day, and average rating deviation.

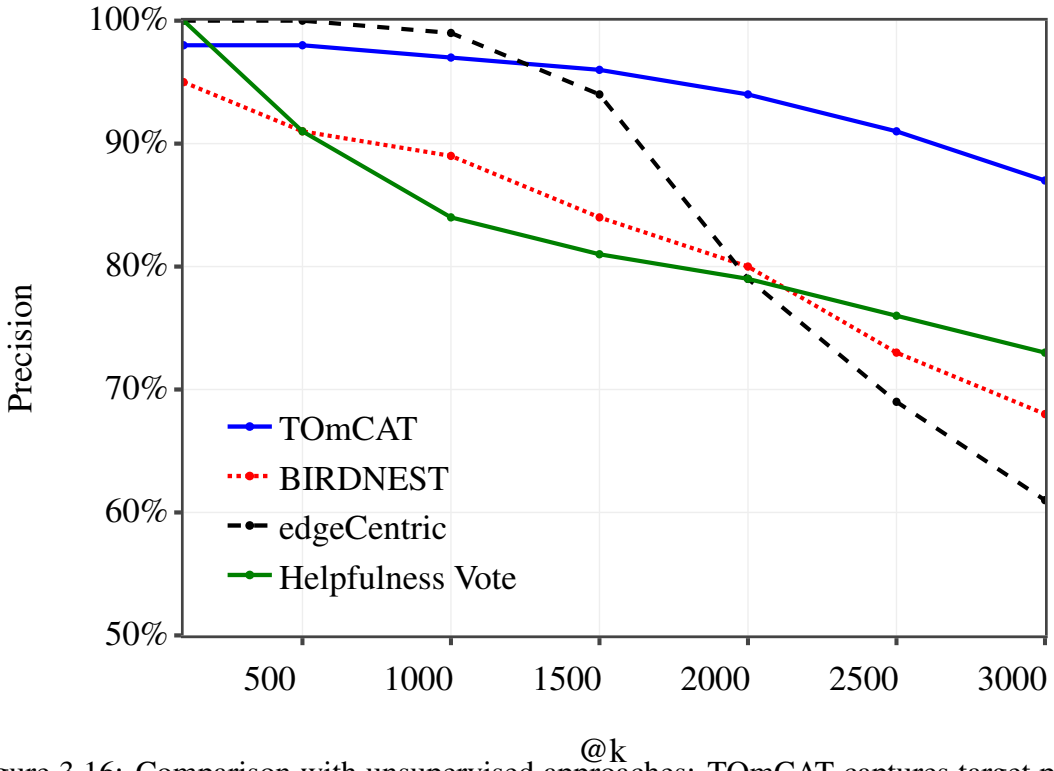


Figure 3.16: Comparison with unsupervised approaches: TOmCAT captures target products with higher precision

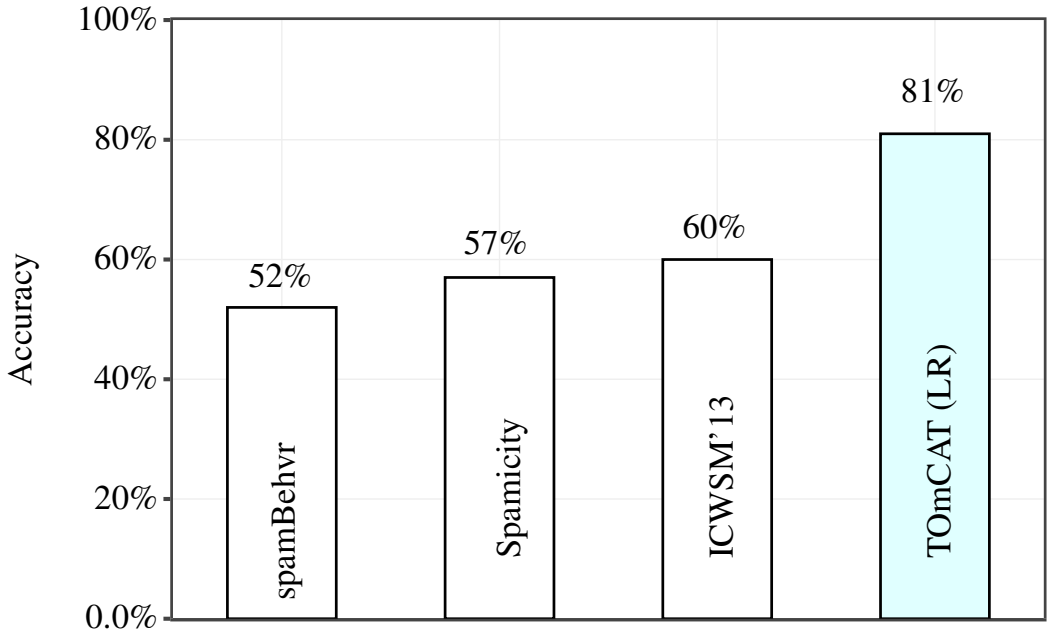


Figure 3.17: Comparison with supervised approaches: TOmCAT captures target products with higher accuracy

For a fair comparison, we applied Logistic Regression (LR) as the classifier used in these baselines on corresponding feature sets. We report accuracy as the appropriate metric for balanced datasets. As shown in Figure 3.17, TOmCAT outperforms its alternatives significantly with 81% accuracy.

We can conclude that baselines that are originally targeted at the reviewer level may miss some of the subtle behaviors evidenced at the product level. Also, they tend to capture users with clear patterns of spam-behavior while target products do not show such clear anomalous signals and they contain mixed legitimate and non-legitimate behaviors as they receive reviews from actual users as well.

Table 3.7: Alternative Datasets Description

	# Items	# Reviews
Amazon [19]	13,449	1,573,555
App store	2,858	304,450
Yelp	174,567	5,261,669

3.3.3.2 Evaluation Over Other Datasets

Here, we evaluate the effectiveness of the proposed approach on three other datasets: from Yelp, the App store, and another existing Amazon dataset [19]. The goal is to detect manipulation patterns left by crowd attacks at the target level using only timing and sequence of ratings. Hence, detection at either reviews or reviewers level which aims to investigate textual content or network behaviors are ruled out. Table 3.7 shows the summary of each dataset in terms of number of items and number of reviews. It should be noted that we re-apply the feature extraction methodology and build the base-model over these datasets. Since the main challenge here is lack of ground truth, we use complementary information e.g., an item could be a target if it received reviews from suspicious reviewers, to evaluate the results.

Amazon [19]: This dataset has been used for review spam detection since 2007. It includes information about reviewers as well which we use as the support information to evaluate our model. we filter out products with fewer than 50 reviews and end up with 13,449 items. **Findings:** Ranking items by their anomaly scores (Sigmoid values), we find significant number of items with Sigmoid score of 1 (~2,000 items) meaning they are extremely similar to our ground truth. For evaluation, we pulled out 100 of these items randomly and investigated their reviewers. The idea is that if they have a review written by a suspicious reviewer, then they are potentially a target of manipulation. We follow the same method to identify suspicious reviewers as explained in section *Identifying Suspicious Reviewers*. We observe that **95 out of 100** top-ranked items by our approach do indeed receive at least one review from suspicious reviewers.

App store: This dataset has been introduced in [96] where it follows similar strategy described

in this dissertation to identify target mobile Apps that includes 100 primary target Apps and in total 2,858 Apps. **Findings:** Evaluating 100 of top-ranked Apps, we find *47 of them are subset of primary target Apps*. we also investigated reviewers associated with these Apps and observed **86 out 100** Apps received at least one review from suspicious reviewers.

Yelp: This dataset is released as part of round 11 of the Yelp challenge in January 2018 [3]. **Findings:** Interestingly, ~19,000 items are ranked top with anomaly score of 1. We pulled out 100 of the top items randomly for further evaluation. Unfortunately, the Yelp dataset only provides information about items not reviewers, so previous method to identify suspicious reviewers does not apply here. Therefore, we explored other available meta-data to support our findings. For example, **16 out of 100** businesses are closed now and **59 and 79 of the items** received fewer than 10 and 20 helpfulness votes respectively while on average items on Yelp receives 42 helpfulness votes.

In summary, TOmCAT performs well in uncovering target products in our original Amazon dataset. This framework is also extrapolated on other review platforms. Furthermore, TOmCAT outperforms the baselines. Despite its success in addressing *existing crowd attacks*, it may perform poorly against strategic attackers who aim to nullify timing-based features. This motivates us to build upon TOmCAT and propose TOmCATSeq complementary approach benefiting from minimal features.

Table 3.8: TOmCATSeq Performance Evaluation

Target Products			Random Products			
Recall	**Precision**	**F1**	**Recall**	**Precision**	**F1**	**Accuracy**
94	74	83	63	90	74	**79**

[3] https://www.yelp.com/dataset/challenge

3.3.4 Complementary TOmCATSeq

As manipulators are constantly evolving their strategy to circumvent new detection methods, the core TOmCAT footprints may lose their power against potential hard-to-detect attack behaviors. Indeed, we find empirically that inter-arrival time plays an important role in detecting manipulation in our dataset. But what if attackers undermine the power of these footprints?
To test this attack strategy, we consider TOmCAT without any temporal features (keeping all other configurations the same). As reported in Table 3.6(w/o Temporal Features) there is a significant drop across various performance metrics. For example, recall/precision of target products and overall accuracy drop from 86/83/84 to 70/69/69 respectively. This indicates the importance of careful consideration of strategic attacks as the TOmCATSeq strive to address such scenarios.

Costs Borne By Attackers. This suggests that attackers may be able to subvert TOmCAT, though at some cost. For example, reviews launched by crowd attacks typically show up in a short time window. However, prolonging inter-arrival time between fake reviews to mimic base-model distributions and conceal anomaly patterns may affect the ultimate goal of crowd attacks in several ways: (i) *Slowing Down the Recovery Period:* As there is a delay between fake reviews, the overall rating affected by negative ratings will be rebuilt slowly, meaning that potential customers may be discouraged by the low overall rating. (ii) *Receiving Legitimate Low Ratings:* Since fake reviews are posted at a slower rate to avoid detection, other legitimate low ratings have an opportunity to arrive, diminishing the impact of the crowd attack.

TOmCATSeq Structure. Regardless of the negative consequences to attackers, such strategic attacks pose serious challenges to the ongoing success of TOmCAT and models built on similar crowd footprints.The main idea of *TOmCATSeq* is to exploit only rating patterns via a novel Rating-based Bidirectional LSTM model. It has two main advantages: First, since it is an end-to-end model, it avoids the need for carefully designed features, instead learning effective representations directly from the input sequence data. Second, it only focuses on rating behavior regardless of their temporal characteristics, meaning that attacks on the timing of ratings are powerless.

Long Short-Term Memory networks known as LSTM is special type of recurrent neural networks (RNN) introduced in [97]. The chain-like nature of RNN naturally fits time-series data as in our ratings scenario. Despite the advances in language modeling, speech recognition, machine translation and computer vision using RNN, the power of deep learning in detecting review manipulation has not been explored yet. In a nutshell, RNNs at each time-step i transfer the corresponding input x_i along with the information obtained from previous hidden state h_i to a new hidden state h_{i+1}. LSTM is popular form of RNNs due to its capability in remembering long-term dependencies. Traditional RNN-based models do not perform well on long sequences since they only reintroduce information a few steps back from the current step. LSTM solves this problem by designing a more sophisticated hidden states.

Traditionally, LSTM is used for text data in which the input at each time-step is a one-hot vector of the corresponding word in the sequence. In rating data, we only have 5 possible different ratings/words so, the vocabulary size is very efficient while that of language models has more than 10K words [98].

We also leverage bidirectional LSTM which provides the capability of taking information from both earlier and later in the sequence. Crowd attacks properties motivate the practice of bidirectional LSTM in our problem. To figure out the occurrence of a manipulation attack, it is not sufficient to just investigate the first part of the sequence, since the model requires more information than just observing one or more low-rate reviews as an attack trigger. Thus, information from the other direction of the sequence containing a series of fake reviews is also required. For this purpose, bidirectional LSTM adds a backward recurrent layer. Hence, TOmCATSeq can detect abnormal rating patterns by memorizing events from past and future time-steps.

The input to the network is the rating sequence of products representing each rating value as a one-hot vector. We treat 1 and 2 stars similarly so the vector representation of each rating has only 4 dimensions. We tried with 5 dimensional vectors and results are similar or slightly worse. we can relate this to the fact that ratings 1 and 2 are treated equally as negative feedback.

TOmCATSeq comprises three layers as *Embedding layer*, *Bi-LSTM layer*, and *fully connected*

layer. The Embedding layer ecodes the input one-hot vectors into embedding vectors. The output of the last time-step in the Bi-LSTM layer is fed into the fully connected layer with Sigmoid as activation function. Furthermore, to prevent over-fitting, we use *dropout* [99] regularization technique. In training stage, dropout sets a portion of Bi-LSTM hidden units to zero with a probability determined by dropout rate.

TOmCATSeq is a binary classifier that learns to classify a rating sequence belongs to a specific product as target or non-target. More formally, the goal is to learn a classification function $f(R_p, \theta)$ that determines the label of product p, given a set of model parameters θ. In training stage, the model is fed training instances (R_p, l_p), where R_p is rating sequence of product p and l_p is the actual label acquired from the ground truth. We consider the binary cross entropy [93] as loss function.

3.3.4.1 TOmCATSeq Evaluation

Here, we present the evaluation results of TOmCATSeq.

Experimental setup: we build TOmCATSeq using the Keras framework [100]. The model hyper-parameters are tuned using grid search. The number of epochs, the size of batches, the size of embedding layer, and the hidden size of BiLSTM are selected from [5, 6, 7, 8, 9, 10], [4, 8, 16, 32], [8, 16, 32, 64], and [8, 16, 32, 64] respectively. The learning rate and the dropout rate are selected from [0.01, 0.001, 0.0001] and [0.0, 0.2, 0.5, 0.7, 0.9] respectively. The parameters of the network are optimized by employing Adam optimizer [101]. The weights are initialized based on various number of distributions as *uniform, normal and zero*. We set the maximum sequence length to be 150 empirically. Therefore, in the training stage we limit ourselves to items with at most 150 ratings and address shorter ones with zero padding. In the inference stage, sequences longer than 150 ratings are chunked into multiple sequences. However, sequential chunks are not mutually exclusive but they have some overlap. If that were the case we would miss the crowd attacks happening on the edge of two chunks because their ratings are divided between two different sequences. To overcome such trivial but destructive cases, we set the overlap to be 20 ratings (as crowd campaign size is typically between 5 to 10, it assures to cover crowd attacks as many as

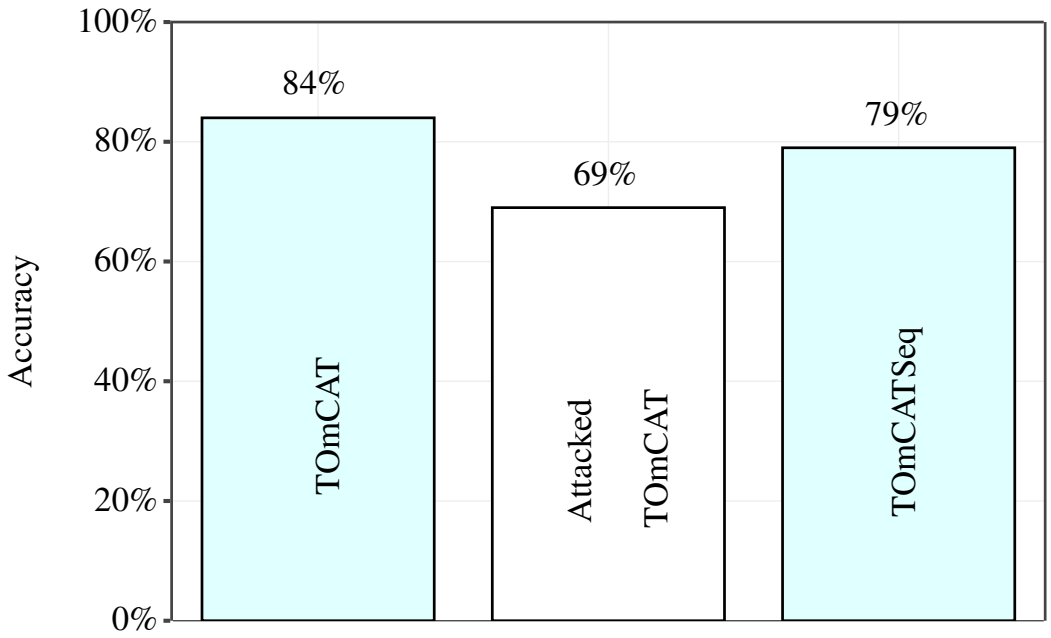


Figure 3.18: Impact of Removing Temporal Features on TOmCAT and TOmCATSeq Performance

possible). In the evaluation stage, if at least one chunk associated to a product is recognized as a target then the corresponding product is considered as a target product.

Results: we report the evaluation results on testing data in Table 3.8. Briefly, 94% of target products have been captured by TOmCATSeq and 74% of products that are recognized manipulated are actual target products. Putting it all together, Figure 3.18 demonstrates the performance of TOmCAT (84%) and how it is affected (69%) when we model attack footprints using only ratings data. Finally, TOmCATSeq is only 5% worse than when we have all the features and is able to recover the accuracy (from 69% to 79%) by relying on an end-to-end RNN-based model fed only rating sequences.

Discussion: The initial evaluation of TOmCATSeq shows promising results relying on rating data while temporal features are left out. This reinforces the capability of TOmCATSeq in the face

of strategic attackers who may adopt a normal temporal distribution to circumvent the detection model. It tells us that even if the attack footprints are not explicit enough to be formulated as hand-crafted features, TOmCATSeq which is built over BiLSTM can perceive hidden patterns left by crowd attacks. This also demonstrates the value of our ground truth which can leverage the power of RNN-based models while unsupervised approaches lack the capability to uncover *difficult-to-detect* review manipulation. However, TOmCATSeq has lower precision compared to TOmCAT (74% v.s. 83%) meaning it misclassifies some non-target products. Of course, strategic attackers may seek to undermine approaches like TOmCATSeq, so we are engaged in a continuing effort to defend against future attacks. For example, adaptive attackers [102] can still circumvent TOmCATSeq by following legitimate rating behaviors.

3.3.5 Conclusion

The TOmCAT framework uncovers crowd review manipulation attacks on sites like Amazon. The proposed model – unlike previous efforts that focus on identifying manipulation at the review or user level – investigates manipulation in the aggregate at the target level. The main contributions are to: (i) model product review behavior through a set of micro and macro features inspired by the presence of restoration and promotion attacks; (ii) evaluate the effectiveness of our TOmCAT model on other online review platforms; and (iii) exploit hidden patterns of manipulation attacks to defeat strategic attackers by leveraging RNN on rating manipulation scenario for the first time. Our results are encouraging, indicating that our model can indeed discover many target products.

4. AI-BASED MANIPULATION

In this chapter, we describe the AI-powered review manipulation strategy which covers the forth and fifth research objectives of this dissertation. As discussed in the previous chapter, large crowd-based manipulation campaigns can spread fake reviews in review systems [8, 7, 6, 103]. In these campaigns, a crowd of workers are paid in exchange for positive reviews, showcasing the potential of marshalling large human workforces to undermine the trustworthiness of online reviews. Since these reviews are written by humans and paymasters typically require them to write realistic reviews – e.g., reviews often have to meet a minimum length and contain positive but not skeptical comments – they often go undetected by modern detection algorithms that focus on review content. Still, crowd campaigns may leave manipulation traces, e.g., as reviews arrive synchronized in time [8] or forming a dense community over the large and mostly sparse co-review graph [7, 35], which can be helpful in their detection.

AI techniques and in particular deep-learning algorithms have become very popular in recent years, leading to new opportunities for attack organizers to leverage the power of AI to spread fake reviews on online review sites rather than using crowd workers.

However, this strategy creates new challenges for both manipulators and defenders. On one hand, generating high quality reviews to be readable by humans is a challenge on its own. On the other hand, AI removes the cost of paying crowd workers and therefore becomes more scalable. It also can control the rate of posting fake reviews, so eliminating the patterns left by crowd campaigns which can be critical to the success of the proposed models for detecting crowd attacks.

This dissertation identifies a new class of attacks that are transferable across different domains leveraging recent advances in transfer learning [67]. We develop a universal model that can easily target new domains for which we have only limited training data, leading to potentially wide-ranging attacks on review systems. The main idea is to develop a universal model from a large collection of reviews (say from Yelp) to capture the general properties of the language used in online reviews and to comprehend the commonly used linguistic patterns.

Based on this model, we conduct a comprehensive empirical study of its effectiveness in generating high-quality fake reviews. We find that machine learning spam detectors cannot distinguish synthetic reviews from real reviews. Furthermore, through a user study, we find that human examiners perceive synthetic reviews as real ones, meaning that new neural generative models of fake reviews pose serious risks to the validity of online review platforms. We compare model-generated reviews with fake reviews written by crowd workers and find that human examiners cannot distinguish between the two. We further compare our approach with state-of-the-art work [4]. Paired with this troubling attack vector, we propose a new defense mechanism that exploits the distributed representation of these reviews to distinguish between real and model-generated ones. This RNN-based discriminator can uncover automated fake reviews with high accuracy. Concretely, the main contributions are:

1. We introduce a new class of attacks on online review platforms using *transfer learning* to automate review generation across different domains. To do this, we propose the framework DIOR for Domain Independent Online Review generation (Section 4.1)

2. We perform a comprehensive empirical study on the robustness of synthetic reviews against traditional spam detectors and human examiners. We show that our proposed framework beats the baselines and competes with crowd written fake reviews (Sections 4.2 and 4.3).

3. We propose a new defense mechanism that leverages the distributed representation of the reviews to detect synthetic reviews with high accuracy (Section 4.5).

4.1 The Proposed DIOR Framework

In seminal work, Yao et al. [4] presented the current state-of-the-art approach to generate fake reviews. In a nutshell, this work leverages neural language models to generate synthetic reviews. The language model is trained over restaurant reviews from Yelp at a character-level granularity. Once trained, the model generates reviews character by character. We refer to this model as *CharLSTM*. CharLSTM generates domain dependent reviews meaning the whole training

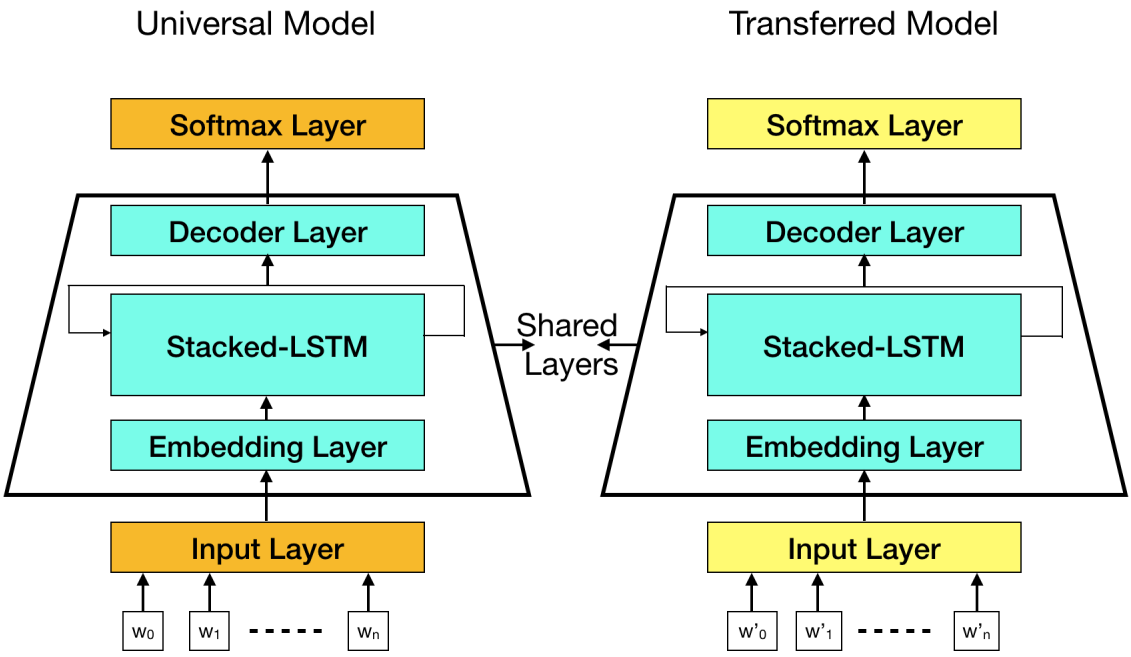


Figure 4.1: DIOR framework to generate domain independent fake reviews.

process needs to be replicated from scratch to generate reviews for any other arbitrary domain (e.g., Mobile accessories on Amazon or Apps on App Store and so on). The proposed approach is not fully automated as it applies a post-processing step (known as customization) which replaces some generated words with more suitable ones. Also, character-level language models need to capture longer dependencies and learn spelling in addition to syntax and semantics, so they are likely to become grammatically more error-prone.

On the other hand, the main advantage of character-level over word-level language modeling is its significantly smaller vocabulary. However, online reviews tend to be short, and centered around a limited range of topics determined by the review domain. For example, the quality of food and service in restaurants on Yelp or the value of a stay reviewed on Airbnb. Therefore, users typically adopt a limited vocabulary ($\sim$30k) compared to a general language domain ($\sim$300k) [104], meaning that word-level models could potentially generate high-quality reviews that avoid traditional traps like misspelling errors.

This dissertation proposes a universal model capable of generating fake reviews at word-level

granularity and is transferable across different domains. The intuition is that users use similar language to write reviews on different domains, e.g., *I enjoyed the food* and *I enjoyed using this App* differ only in domain-dependent vocabularies while the surrounding words express similar semantics. Hence, we can transfer the knowledge obtained during training of the universal model to conduct learning for any desired *target domain* efficiently with possibly smaller review samples as it only needs to adapt to the idiosyncrasies of the target review domain. We consider mobile accessories reviews on Amazon and mobile App reviews on App store as the target domains.

We first present the design of our proposed framework DIOR for Domain Independent Online Review generation (Figure 4.1). DIOR comprises two steps: (i) building the universal model; and (ii) refining to the target domains. The main assumption is to transfer the knowledge from a source domain with large training samples to a target domain efficiently with possibly smaller number of samples. Therefore, we pick Yelp as the source model where significant amount of its reviews are available as described in Table 4.1. We also show empirically how much target training samples are sufficient to build the transferred model.

4.1.1 Background

We begin with a quick refresher on the basics of language models based on recurrent neural networks (RNNs) [105, 106, 107] before diving in to the design of DIOR. RNNs build a "memory" cell [97] to maintain the information about what it has seen from the input sequence so far and transfer the information to the next time step. An RNN cell is composed of a set of high dimensional weights learned during the training stage to capture the dependency among the words in the training samples.

Training stage. At each time step t, the network takes in the current word w_t along with the current hidden state h_t, that encodes the sequence till the time step t, and outputs a distribution over the vocabulary for the next word. The output distribution essentially describes the probability of observing each word w' in the vocabulary given the sequence w(<=t) ($P(w'|w_1, \ldots, w_t)$). The output is then compared with the desired output w_{t+1} that is the next word in the training sequence

through cross-entropy loss defined as follows:

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{|V|} y_{t,j} \log \hat{y}_{t,j} \tag{4.1}$$

where y is one-hot vector wherein the index corresponding to the desired output word w_{t+1} is set to one. $\hat{y}$ is the model probability vector activated by the softmax function that is interpreted as the probability distribution over the words. Both y and $\hat{y}$ are V-dimensional vectors, where V is the size of vocabulary and T denotes the length of the sequence. The network parameters are then updated over multiple iterations to minimize the loss value.

Generating stage. This is an auto-regressive approach, where the trained language model can be used to generate a sequence of words. It begins by taking in the initial hidden state h_0 and word w_0. At each time step t, it takes in the word predicted at $t-1$ along with the hidden state h_{t-1} and predicts the distribution for the next word w_t and also updates the hidden state to h_t. By feeding w_t back to the model, it produces another probability distribution to predict the next word.

Diversity Control. To control the diversity of predicted words, a hyper-parameter called temperature τ is used in the generating stage by scaling the output vector o before applying the softmax function. In other words, the logits in the output vector are divided by τ. The softmax function over o produces probability value $\hat{y}_k$:

$$\hat{y}^k = \frac{e^{o^k/\tau}}{\sum_{j=1}^{V} e^{o^j/\tau}} \tag{4.2}$$

where o^k represents the element of the output vector corresponding to the word at index k in the vocabulary. When τ is set to be 1, the softmax is directly computed on the logits. Lower temperatures result in more conservative predictions since it is less likely to pick from unlikely word candidates. The next word is selected based on a multinomial distribution over $\hat{y}$ probability vector.

4.1.2 Universal Model

Given this background, we turn now to the design of DIOR. We first start by building a universal model over a large collection of user reviews. We use a collection of Yelp reviews. R^Y indicates the concatenation of the all reviews in the training set. Since the language model aims to predict the next word at each time step, the labels are just like the inputs but shifted by one word. For example, if the input sequence is "we ate at this restaurant" then the language model is trained to predict "ate at this restaurant" sequentially given the first word (we). Hence, labels are defined as $L^Y = R^Y[1:]$.

Then, we break down R^Y into sequences with respect to the number of the time steps (T) in the language model. For faster gradient decent update, the resulting sequences are divided into mini batches with respect to the batch size (bs). Therefore, for a given mini batch b, the input to the learning algorithm would be matrix X_b^Y of shape $T \times bs$ where values represent the reviews tokens. The label matrix y_b^Y is built similarly over labels L^Y.

We now describe the architecture of the DIOR at time step t which can be generalized for any time step. It consists of three layers: an Encoder, 3-layer stacked-LSTMs, and Decoder. The encoder layer uses a trainable matrix (W_e^Y) to learn the embedding representation of the input tokens. The matrix shape is defined by vocabulary size V and embedding size em. The embedding for words are obtained by multiplying their one-hot representation with the W_e^Y

$$\xi(X_{b,t}^Y) = W_e^Y \times X_{b,t}^Y$$

From these embedding representations, we employ a stacked-LSTM (3 layers) to model dependency between words.

$$h_t^1 = LSTM(\xi(X_{b,t}^Y), h_{t-1}^1)$$

$$h_t^2 = LSTM(h_t^1, h_{t-1}^2)$$

$$h_t^3 = LSTM(h_t^2, h_{t-1}^3)$$

The Decoder layer is a linear transformation which decodes the output of the last LSTM layer (h_t^3). However, we use the “Weight Tying" technique that allows sharing of weights between encoder and decoder layers to reduce the number of learned parameters [108]. Therefore, h_t^3 is decoded using the transpose of the embedding matrix (W_e^Y) and the result vector is activated by the softmax function to produce the probability distribution over the words in the vocabulary. It should be noted that the hidden size of the last LSTM layer is set to be equal to the embedding size em.

$$\hat{y} = softmax(h_t^3 \times (W_e^Y)^\tau)$$

The output $\hat{y}$ is then compared with desired labels $y_{b,t}^Y$ through cross entropy loss as defined in Equation 4.1 and model parameters are updated accordingly during multiple iterations.

In addition to this architecture, we apply two regularization techniques introduced in [67] to capture the dependency between words in the language more effectively. These techniques are centered around fine-tuning the learning rate η i.e., a hyper-parameter that controls how much to update the model parameters (weights) with respect to the loss gradient.

The first technique called *Discriminative Fine-tuning* suggests tuning each layer with different learning rates instead of using a single learning rate through all layers of the model. The intuition is that different layers capture different features [109], so they should be tuned differently. Considering the update process at time step t:

$$\theta_t^Y = \theta_{t-1}^Y - \eta \dot{\nabla}_{\theta^Y} J(\theta^Y)$$

where θ^Y represents the model parameters, and $\nabla_{\theta^Y} J(\theta^Y)$ is the gradient with respect to the cost function. With this technique, the update process at each layer l would be:

$$l(\theta_t^Y) = l(\theta_{t-1}^Y) - \eta^l \dot{\nabla}_{l(\theta^Y)} J(l(\theta^Y))$$

The second technique called *Slanted triangular learning rates (STLR)* suggests that not only we do need to consider different learning rates for different layers but also that we need to change the learning rate through the training iterations rather than using a fixed learning rate. The intuition is that varying the learning rate helps the model converges efficiently. Through this technique, the learning rate first linearly increases and then linearly decays.

After training, the set of learned model parameters θ^Y are used to generate reviews for our universal model. Empirically, we will study Yelp as our source model in the following.

4.1.3 Transferred Model

Given this universal model, we now turn to transferring this model to new domains for which we have only limited training data. In this way, a single learned model can potentially be used to launch attacks against a host of other review systems. For ease of presentation, we assume in this section that the target domain is Amazon and indicate the corresponding notations with Am superscript e.g., θ^{Am} denotes the target model parameters. In practice, the target domain could be any domain for which some reviews can be sampled.

A straightforward approach for transfer learning is to focus on the model's first encoder layer and initialize the weights with pre-trained word-embeddings [110, 111] or embedding matrix W_e^Y learned for Yelp vocabularies in our case. However, this approach still trains the target language model from scratch and treats pre-trained word embeddings as fixed parameters.

A recently introduced approach [67] proposes to transfer the knowledge from *all the layers* to benefit from source model to the fullest. Hence, we use the same architecture as universal model comprising an encoder, 3-layer LSTMs, and decoder and initialize the refined model parameters θ^{Am} with the parameters of the universal model θ^Y.

Initialization. The parameters of the Yelp generative model (θ^Y) are reused as the starting point for the Amazon generative model. However, the vocabularies are not the same in the two domains,

Table 4.1: Summary of Review Datasets

	# Revs	# Tokens	Vocab	Med rev Len	Avg rev Len	Training	Validation
Yelp	318,392	31,514,567	35,394	69	100	286,552	31,840
Amazon	108,664	10,633,295	20,731	49	97	97,797	10,867
App store	231,199	12,131,926	24,614	41	53	208,079	23,120

so the universal embedding matrix W_e^Y with the shape of $|V^Y| \times |em|$ cannot be used directly because the size of the transferred model's embedding matrix W_e^{Am} needs to be $|V^{Am}| \times |em|$. The average over embedding of the common words in the two domains is used to initialize the unseen words e.g., Amazon domain dependent words.

$$m = Avg(w_e^i \forall w^i \in V^Y \wedge w^i \in V^{Am})$$

$$W_e^{Am} = ||w_e^i \forall w^i \in V^Y \wedge w^i \in V^{Am}|| \ldots ||m \forall w^i \in V^{Am} \wedge w^i \notin V^Y||$$

After initialization, the target model is trained using the same set of techniques introduced in Section 4.1.2. Together, this DIOR model design promises the potential of generating high-quality reviews at word-level granularity in an inductive transfer-learning framework.

4.2 Experimental Design

Before turning to our empirical study of the proposed DIOR framework, we describe the experimental design. we consider Yelp and Amazon/App Store reviews as the source and target domains respectively. Each review contains the text of the review, and the rating score in the range of one to five stars. we present the results based on reviews with positive sentiment, so we keep only the reviews with 5-star ratings. we split the datasets into training and validation sets with ratio of 90% and 10%. Four disjoint datasets are used for generating and evaluating synthetic reviews. Table 4.1 summarize the statistics on these datasets.

Yelp. we use the Yelp Challenge Dataset (round 11) released in January 2018.[1] This dataset

[1] https://www.yelp.com/dataset

Table 4.2: Example of the synthetic five-star reviews at different temperatures.

Temperature	**Generated Reviews (Yelp)**
0.2	I love this place ! we 've been here several times and we 've never been disappointed . The food is always fresh and delicious . The service is always friendly and attentive . we 've been here several times and have never been disappointed .
0.4	I 've been to this location twice now and both times we 've been very impressed . we 've tried their specialty pizzas and they 're all really good . The only problem is that they 're not open on sundays . They 're not open on sundays .
0.8	I 've eaten here about 8 times . we 've been introduced to this place . Its always busy and their food is consistently great . we LOVE their food , hence the name . It is so clean , the staff is so friendly , and the food is great . we especially like the chicken pad thai , volcano roll , and the yellow curry .
Temperature	**Generated Reviews (Amazon)**
0.2	I have been using this case for a few weeks now and we love it ! we have had this case for about a month now and it is still holding up great ! we have dropped my phone a few times and the case has protected it perfectly ! we would recommend this case to anyone !
0.4	after reading the reviews we read the reviews and decided to give it a shot . we am very pleased with the results . The quality is great , it fits perfectly and we do n't have any problems with it . It 's a great value for the price .
0.8	the case works great ! it has a soft rubber insert that goes over the hard shell . The hard plastic shell has a soft inner shell and the hard case is hard plastic . It is very sticky and has not fallen out or dropped or fallen apart .
Temperature	**Generated Reviews (App store)**
0.2	we love this game so much ! it 's so fun and addicting ! we love the fact that you can play with friends and family !
0.4	this app is great ! we have been using it for years and it has always been reliable and reliable . we have been using it for a long time now and it is always reliable .
0.8	this app is a great tool for discovering new things : being able to search for films and putting reviews on particular items as well as having a way to download stories from the app .

contains $\sim$5m reviews targeting $\sim$174k businesses. we extract reviews corresponding to restaurants and find 318,392 five-star reviews containing 31,514,567 total words and 35,394 unique words, a sufficiently large dataset to serve as our transfer learning source task.

Amazon. This dataset introduced in [84] includes Amazon product reviews across a variety of categories. In our evaluation, we focus on the cell-phone accessories category and extract 108,664 five-star reviews with 10,633,295 total words and 20,731 unique words.

App Store. This dataset contains reviews about mobile applications with 231,199 five-star reviews and 12,131,926 words and 24,614 unique words introduced in [103].

Model-Generated Dataset. This dataset contains the reviews generated by the proposed language model. The synthetic reviews for Yelp are generated directly from the universal model. The Amazon and App store generated reviews are results of transferring the domain from Yelp to the corresponding domain.

Reproducibility. For the sake of generality, we set the hyper-parameters from a singleton set. The language model has an embedding size of 400, 3 layers, and hidden size of 1,150. It applies Adam optimizer with $\beta_1 = 0.7$ and $\beta_2 = 0.99$. The batch size is set to 64 and the base learning rate for fine-tuning is set to 0.004. The number of epochs are tuned on the validation set.

Review Generation. To generate reviews, the initial word represents the beginning of the reviews which we define by a special token $< sor >$. The generation process continues until the model predicts the end of the review identified by a special token $< eor >$ or the sequence length becomes equal to the median length of the reviews in the corresponding dataset. we set to generate reviews at temperatures $[0.2, 0.4, 0.6, 0.8, 1.0]$. Intuitively, generation at low temperatures reinforces the difference between the occurrence probability of the words and reduces the chance of words with lower probability to be predicted. At low temperatures, the model tends to generate sequences commonly seen in the training data, so it generates repetitive patterns. By increasing the temperature, rarer words become visible to the predictor at the cost of grammar mistakes and incoherency. We evaluate quality of the generated reviews at different temperatures and recognize the optimal temperature value. Samples of generated reviews are shown in Table 4.2. It is notable

that generated reviews at temperature 0.2 tend to repeat themselves.

4.3 User Study

In this section, we evaluate the quality of generated reviews from different aspects by conducting a comprehensive user study. Although metrics such as loss and perplexity values are used to evaluate the performance of generative models, the real test is to understand how synthetic reviews are perceived by end users. *First,* we evaluate the quality of reviews generated by DIOR. *Second*, we evaluate the model-generated reviews against fake reviews written by crowd-based review manipulators. *Third*, we compare DIOR with state-of-the-art work [4].

User Study Guidelines. We set up a crowdsourcing user study to examine whether model-generated reviews are convincing to human readers. We post surveys on Amazon Mechanical Turk (AMT) to assess the reviews. To ensure the quality of responses, we follow a set of guidelines. We insert a trivial question into each survey, which asks the Turker to check if a mathematical equation is False or True. This mitigates the risk of blindly answering surveys. Furthermore, we only accept surveys where the Turker dwelled on the survey for some time (5-7 minutes depending on number of questions in the survey). We also restrict our tasks to those who are located in the United States to guarantee English literacy.

4.3.1 DIOR Reviews

The main aim of the first user study is to evaluate the quality of DIOR generated reviews by analyzing how they are perceived by end users. To do this, for each domain, we design 100 surveys each with 10 reviews out of which 5 are real reviews and 5 are synthetic reviews, each generated at one of [0.2, 0.4, 0.6, 0.8, 1.0] temperatures. Each unique survey is assigned to 3 workers (3 HITs[2] per task), giving us a total of 300 surveys. We provide an instruction on top of the survey highlighting two main points; 1) Turkers are tasked to mark each review as either real or fake. And, for those recognized as fake, they are asked to provide their reasoning using keywords like “repetition", “grammar issue" and “nonsense"; 2) it also shows a sample of real reviews to

[2]Human Intelligence Task

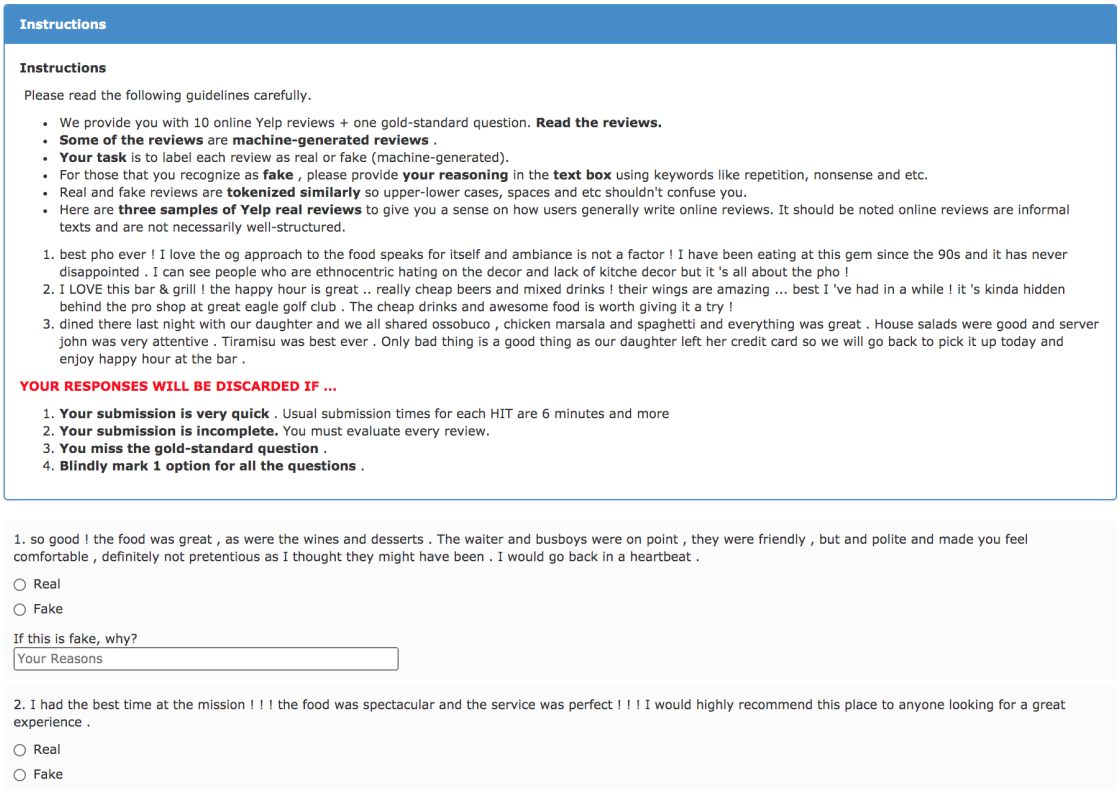


Figure 4.2: Sample of the survey to evaluate quality of DIOR reviews by end users

the Turkers indicating that online reviews are not necessarily well-structured pieces of text. This prevents many real reviews from being marked as fake due to their informal language. An example of the survey is shown in Figure 4.2.

Figure 4.3 demonstrates the performance of synthetic reviews against human judgment at various temperatures. The key point is that such reviews remain quite robust and many of them are recognized as real. Under the best configuration in Yelp, i.e., temperatures 0.6 and 0.8 the percentage of reviews flagged as real is 76.42% and 78.42% respectively. At the same time 89.52% of real reviews are labeled correctly showing a 10% error while they are supposed to pass the human test perfectly. We can relate this to the fact that user-written reviews could be also error-prone on any basis making it a challenge to humans to distinguish between these two types of reviews. In addition, similar to algorithmic evaluation, the DIOR performance improves with the increase in the temperature.

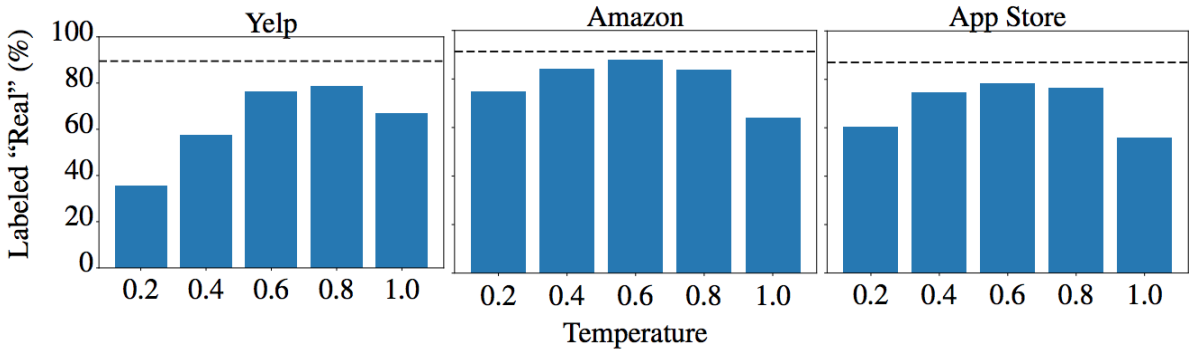


Figure 4.3: Majority of the synthetic reviews at temperatures 0.6 and 0.8 are recognized as real. Dot lines indicate percentage of real reviews that were labeled as real by humans.

By examining Turkers' reasoning while they flag a review as synthetic, we find that many reviews at low temperatures e.g., 0.2 and 0.4 are identified as fake due to repetition and they sound robotic even though they are grammatically correct. On the other hand, the performance of such reviews improves at higher temperatures.

However, we can observe a downfall of the accuracy at temperature 1.0 as examiners believe the reviews may not be coherent and contain a nonsensical argument. We can recognize the best performance occurs at temperature 0.8.A similar pattern is observable in both other domains. In summary, Reviews generated at this temperature can fool human readers and go undetected. And, Human readers are more sensitive to repetition errors than they are to small grammar mistakes.

Hence, in the following studies (Sections 4.3.2, 4.3.3 and 4.4.1), we focus on reviews generated at temperature 0.8 as it is demonstrated to be the optimal threshold for generating reliable reviews understood by humans.

4.3.2 DIOR versus Crowd Manipulators

The first major thrust of this dissertation is to investigate crowd-based manipulation. Here, the research objective is to compare the power of AI-based manipulation with that of crowd enabled attacks. As a reminder, in crowd-based manipulation, a crowd of workers are paid in exchange of positive reviews. Here we aim to evaluate how end users perceive the fake reviews generated by proposed DIOR approach compared with those written by crowd manipulators. This study answers

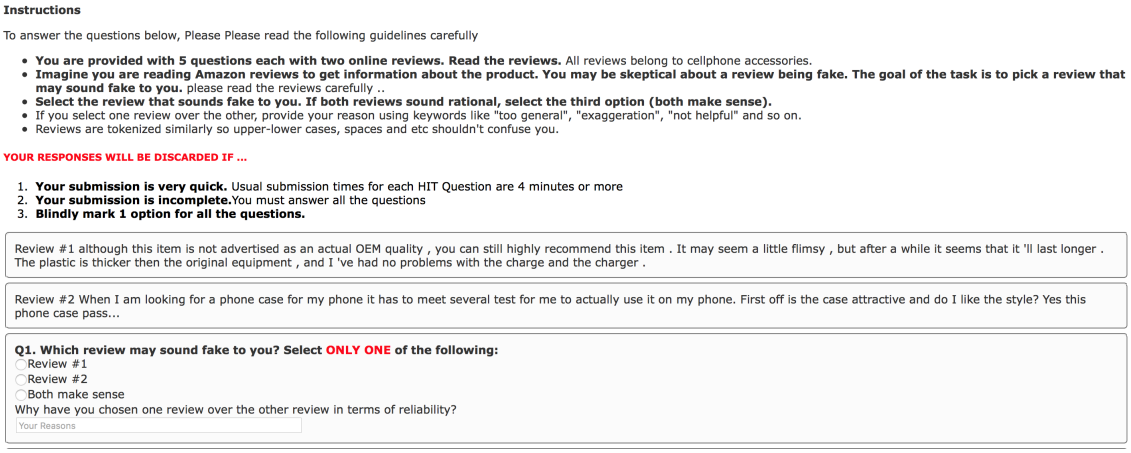


Figure 4.4: Sample of the survey to evaluate quality of DIOR reviews versus crowd written reviews

the important question of whether AI is capable enough to take the place of the crowd campaigns?

We first prepare a ground-truth of crowd written fake reviews. This dissertation introduces a dataset of crowd-based manipulation traces on Amazon [7]. However, the focus of crowd-based manipulation thrust is to detect manipulators and targets of manipulation not to identify individual reviews as fake or real. Therefore, we label reviews as fake using the notion of *self-plagiarism* [8] in which a manipulators simply duplicate a single review on two different products. From this pool of fake reviews, we filter out reviews about mobile accessories for the fair comparison with the Amazon synthetic reviews.

Second, we design 20 surveys each with five pair of reviews each generated by either crowd manipulators or DIOR framework. Each unique survey is assigned to 3 Turkers (3 HITs per task), giving us a total of 300 ($20 \times 5 \times 3$) pairs of reviews. They are tasked to select which of the two reviews sound fake to them or none if they find them to be equally reliable. An example of the survey is shown in Figure 4.4.

As Figure 4.5 shows, both crowd and model generated reviews are equally likely to be detected by human evaluators as fake. For example, 32% and 31% of the answers found crowd written fake reviews and reviews generated by DIOR as suspicious respectively while 37% of the answers found both type of the reviews equally reliable. In summary, end users find reviews generated by

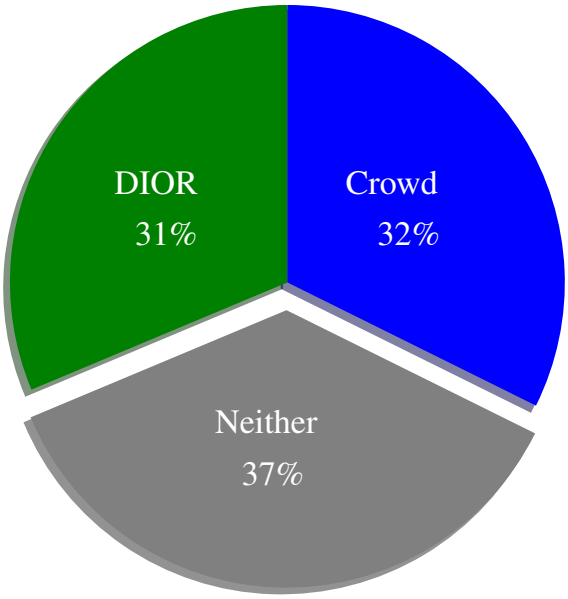


Figure 4.5: Users perceive DIOR generated reviews as reliable as crowd written fake reviews.

DIOR as reliable as fake reviews written by manipulation campaigns.

4.3.3 DIOR versus the state-of-the-art Model

This section compares the effectiveness of the proposed DIOR model to the state-of-the-art work [4]. In summary, CharLSTM is a two-layer character-based LSTM trained over Yelp challenge dataset and generates restaurant reviews. However, the implementation code or a sample of fake reviews generated by this model is not publicly available. Hence, we replicate the model as closely as we could based on the configurations reported in the paper (Section 3.2 Training Process).

Now we conduct a user-study and design 20 surveys each with 20 reviews out of which 12 are real reviews, 4 are fake reviews generated by CharLSTM and 4 are fake reviews generated by DIOR. Each unique survey is assigned to 3 Turkers (3 HITs per task), giving us a total of 60 surveys and 1200 ($20 \times 20 \times 3$) reviews. Each participant is tasked to label four reviews as fake.

On average, the detection rate (recall) is 28% and 45% for reviews generated from DIOR and CharLSTM respectively. Figure 4.6 shows that DIOR generated reviews go undetected with higher probability (72% versus 55%). To conclude, DIOR beats the state-of-the-art model in generating

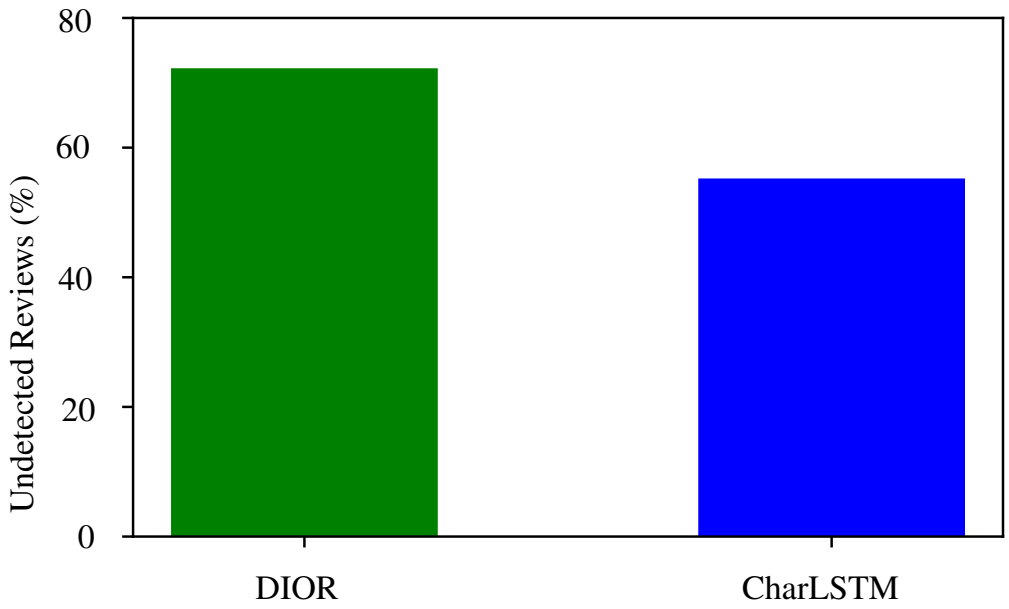


Figure 4.6: Comparison with baseline.

effective reviews.

4.4 Impact of Transfer Learning

In this section, we investigate the impact of transfer learning from performance and resource efficiency points of views. First, how it improves the quality of synthetic reviews compared to individual models. Second, how much data is needed to refine the universal model. Overall, we aim to show advanced language models make it easier to manipulate review platforms.

4.4.1 Performance: Transferred versus Individual Models

In this study, we evaluate the quality of reviews generated from the transferred model against reviews generated from a language model trained from scratch – what we refer to as the individual model for our target domains. To do this, we set up a pair-wise comparison based on a survey-based user study to see how natural reviews generated by two models sound to human readers.

For each target domain, we design five surveys each with five pair of reviews. Each unique survey is assigned to 10 Turkers, giving us a total of 50 surveys and 250 pairs of reviews. The task is to select which of the two reviews sound more natural or both if they find them to be equally natural. An example of the survey is shown in Figure 4.7.

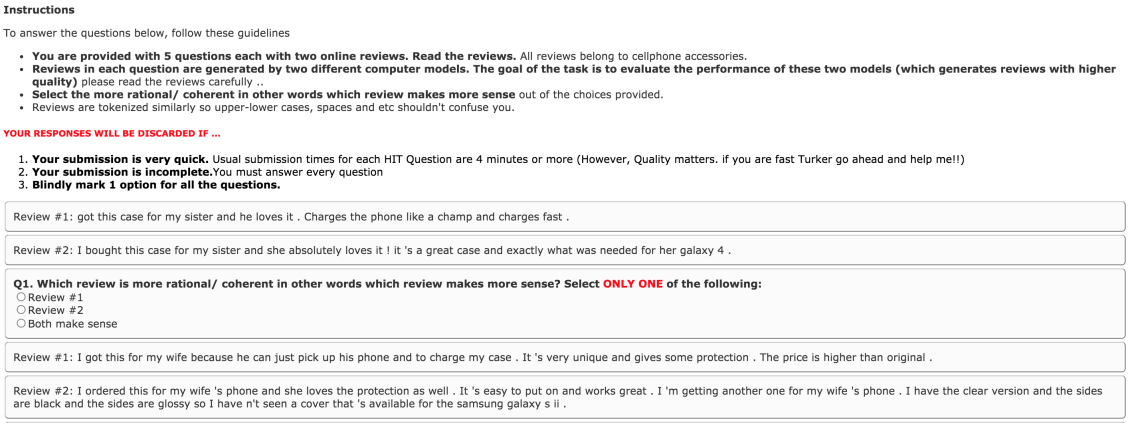


Figure 4.7: Sample of the survey to evaluate quality of reviews generated by transferred and individual models

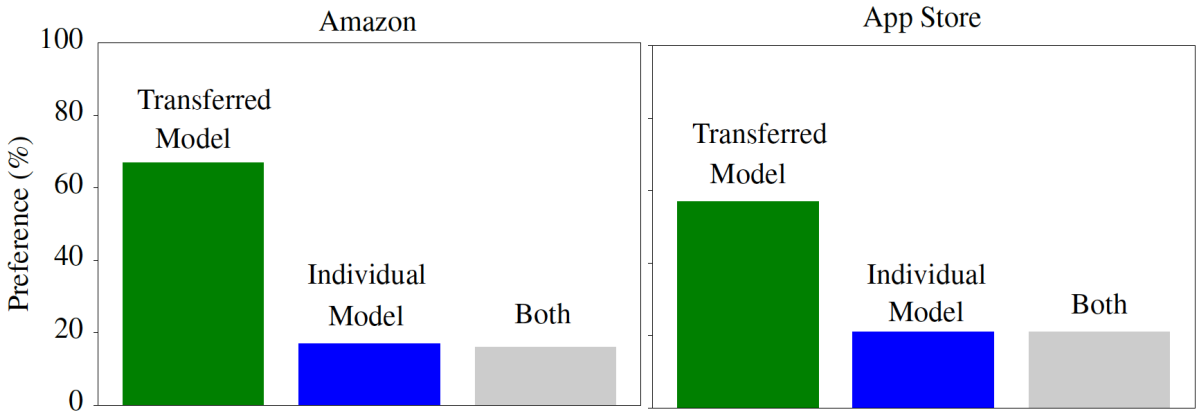


Figure 4.8: Users find reviews generated by transferred models more convincing than those generated by individual models

To ensure that users evaluate reviews based on only the language of the review, we pair reviews with a similar topic for each comparison. For example, both reviews talk about fitness applications in the App domain or headsets in the Amazon domain. As Figure 4.8 shows, for the App store, human readers found 57% of the reviews generated from the transferred model to sound natural as opposed to only 21% of reviews generated by individual model. 21% of responses found both reviews to sound equally natural. We find similar results for Amazon. To conclude, using transfer learning not only facilitate the domain shift but also improves the performance significantly.

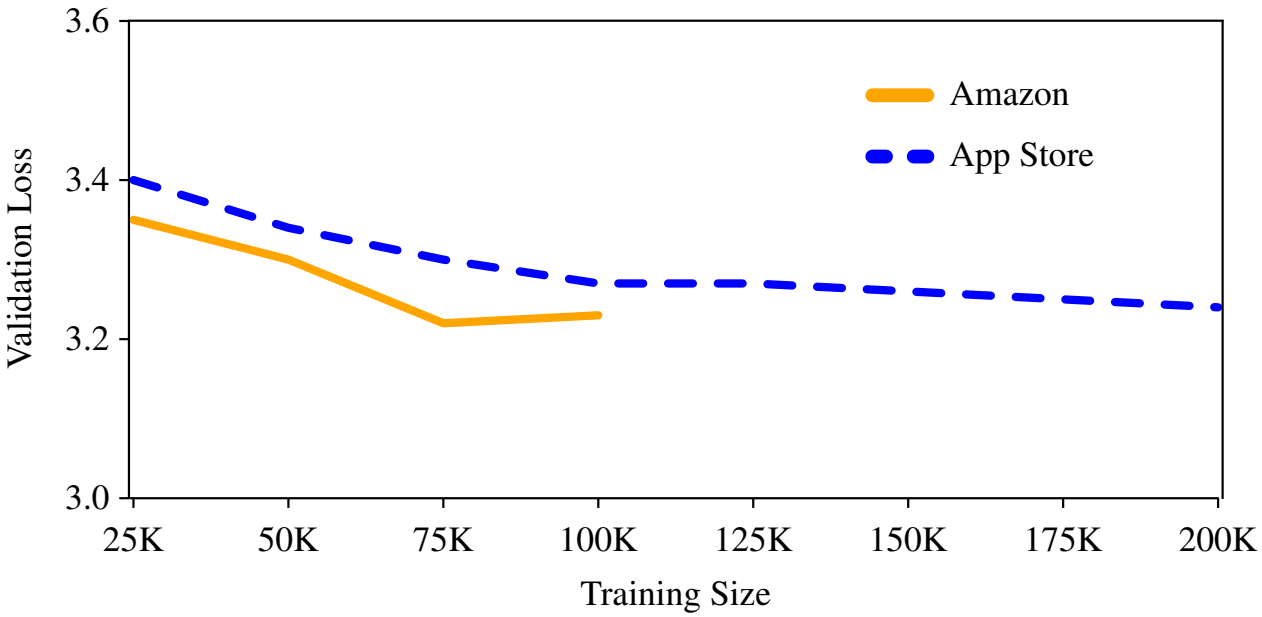


Figure 4.9: The transferred models need reasonably low number of samples to reach stable performance.

4.4.2 Resource Efficiency: Training Size

It is critical to understand the resources needed to refine the universal model. In particular, when the target domain's dataset is not sufficiently large we are eager to know how many reviews are required for the model to converge. we plot the validation loss versus training size for Amazon and App transferred models in Figure 4.9. We gradually increase the training size starting with 25k reviews. According to this figure, we need approximately 100k and 75k training reviews for App and Amazon domains respectively to converge the loss values and achieve a relatively stable performance. In the Amazon domain, the model converges with a smaller set of reviews and we can relate this trend to the size of the vocabulary. According to Table 4.1, there are about 20k unique words in Amazon (less than that of App store with 24k unique words) which helps the model to adapt to the domain more efficiently. To conclude, the transferred models need reasonably low number of samples compared to universal model to reach stable performance.

In summary, we showed that automating generation of fake reviews can expose serious threats to review platforms through different studies. However, despite the success of DIOR in generating

Table 4.3: Performance of spam detector. From temperature 0.8 synthetic and real reviews become indistinguishable

	Yelp					Amazon					App store				
Temp	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**
Acc (%)	92	81	73	64	55	92	82	75	61	56	91	77	69	62	54
Prec (%)	93	82	74	65	56	93	83	76	64	57	92	80	71	62	55
Rec (%)	93	82	74	65	55	92	82	75	61	56	91	78	70	62	54
F1 (%)	93	82	74	65	54	91	82	75	60	54	91	77	69	62	53

realistic reviews, in the next section, we propose a new defense mechanism to combat this threat.

4.5 Proposed Discriminator for Detection

Although model-generated reviews successfully pass the human tests, the question is whether RNN-based language models manage to model the real review distribution? To answer this question, we propose a discriminator capable of distinguishing synthetic reviews from real ones using the underlying distributed representation of the review words learned during the training process of the language model. Before turning into describing the discriminator, we first develop a textual classifier that is the base model to detect spam text as the baseline to evaluate the performance of the discriminator.

4.5.1 Baseline: Spam Detector

Here we develop a machine-learning scheme to evaluate if model-generated reviews carry different linguistic patterns from real reviews. Text classification has been widely used to detect opinion spam on the web [19, 29, 38]. We describe four groups of linguistic features consisting of 12 features in total, following the approach proposed in [4] to classify the reviews.

Similarity feature (1): Measures the inter-sentence similarity within a review. It computes the cosine-similarity between each pair of sentences based on their unigram tokens and considers the maximum value as the similarity feature [112].

Structural features (2): Captures the structural aspects of a review including the average sentences length measured by their number of words and the average word length measured by their

number of characters [30].

Syntactic features (5): Defines linguistics properties of a review based on parts-of-speech (POS) tagging process. It includes percentage of nouns, verbs, adjectives, adverbs, and pronouns [30].

Semantic features (4): Captures the sentiment and subjectivity of a review including percentage of positive, negative, subjective, and objective words. we use SentiWordNet library [113] to extract this type of feature.

For each domain and at each of the temperatures, we sample 10k model-generated reviews and 10k real reviews, for a total of 20k reviews. we split this data into training and testing sets with a ratio of 70% and 30% respectively. we train an SVM classifier with rbf kernel and c=1 (obtained from [0.001, 0.01, 0.1, 1, 10] set through grid-search). After training over all 12 linguistic features, we test the performance of the classifier over the testing data. Results are averages of 10 runs. It should be noted that we tried different classifiers such as Logistic Regression and Linear SVM and observe similar results. Hence, we report the results obtained from SVM as representative standard machine learning classifier.

The evaluation metrics are average of precision, recall and F1-score over both classes of reviews and overall accuracy. Table 4.3 reports the performance of the classifier at different temperatures across different domains. We observe that text classification shows high detection performance at lower temperatures. For example, at temperature 0.2 it classifies reviews with 0.92, 0.92, and 0.91 accuracy in Yelp, Amazon, and the App store domains respectively. We can relate this trend to the fact that reviews generated at lower temperatures tend to repeat themselves (Table 4.2), which makes features like inter-sentence similarity an informative feature to distinguish two class of reviews.

However, we aim to evaluate the performance at temperature 0.8 as we find this to be an optimal temperature in a qualitative analysis by human readers (Section 4.3.1). We observe that model-generated reviews can circumvent the linguistic-based test when an appropriate temperature is selected. Low evaluation metrics at temperature 0.8 – e.g., 0.64, 0.61 and 0.62 accuracy across Yelp, Amazon, and the App store respectively – indicate synthetic reviews do not resemble spam

behavior. In summary, the linguistic-based spam detector may not distinguish synthetic reviews from real reviews.

4.5.2 Discriminator

The key insight is that language models predict the next word conditioned on previously seen words while humans are not restricted by this requirement when they write online reviews. Figure 4.10 visualizes the review embeddings over 400 Yelp review samples in two dimensions using t-SNE [114], with markers corresponding to synthetic and real reviews. Note that the review embedding is computed by composing their word embeddings. we adapt a straightforward approach that represents a review by taking the average over its word embeddings [86]. As we can see from Figure 4.10, synthetic reviews tend to cluster together in the embedding space in particular at low temperatures.

This motivates to explore the manner of words appearing in a sequence in synthetic and real reviews. The proposed discriminator learns a classifier $M(R, \theta)$ that classifies a review R as real or synthetic given its terms and model parameters θ. A bipartite training sample consists of (i) embedding representation of review terms $\{\xi(w_1), ..., \xi(w_{|R|})\}$ extracted from embedding matrix W_e (Section 4.1); and (ii) its corresponding label y i.e., 1 or 0 indicating synthetic and real reviews respectively. For a given batch of training samples b, the loss function based on cross-entropy is defined as follows:

$$J(b; \theta) = \frac{1}{|b|} \sum_{i=1}^{|b|} y_i \log(M(R_i, \theta)) + (1 - y_i) \log(1 - M(R_i, \theta))$$

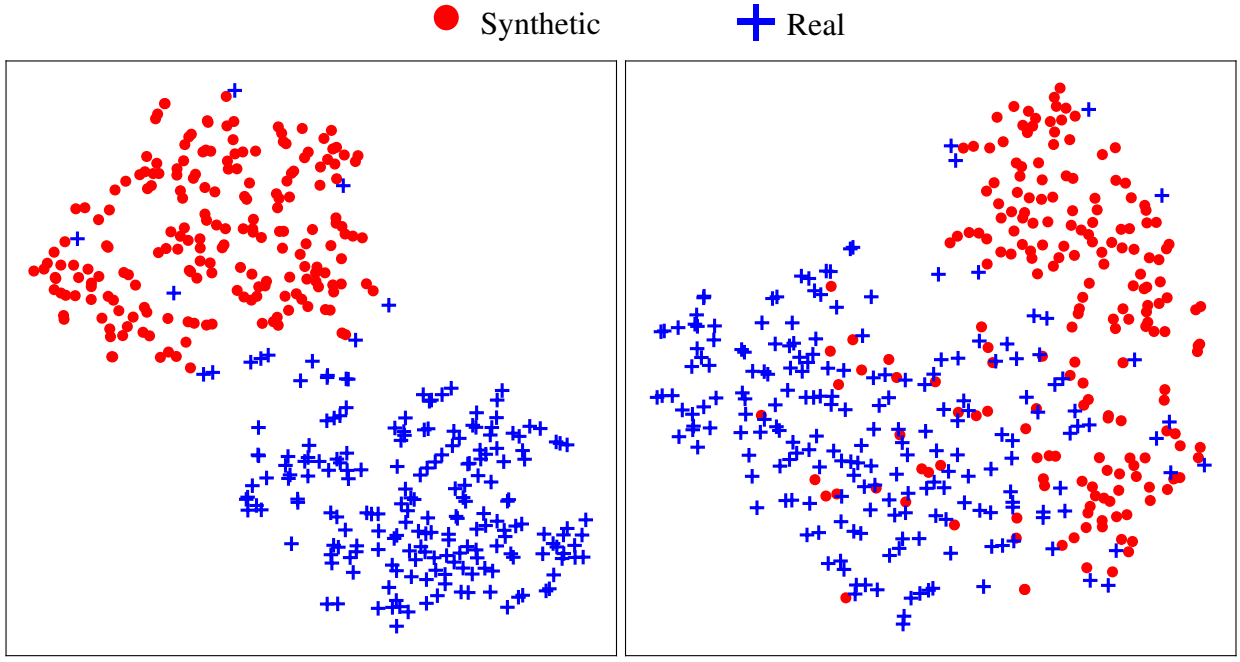
Where $M(R_i, \theta)$ and y_i indicate the predicted and actual labels of review R_i respectively.

Network Architecture. we opt for a straightforward LSTM network which is composed of: LSTM and linear layers. The LSTM is fed distributed representations of review terms (term embedding for short) and the output of the last hidden state is fed to the linear layer where it is activated by the softmax function and outputs the classification result.

Experimental Setup. For each domain and at each temperature, we sample 10k model-generated reviews and 10k real reviews from the training dataset, in total 20k reviews. we split the data into

training, validation, and testing datasets with the ratio of 80%, 10%, and 10% respectively.

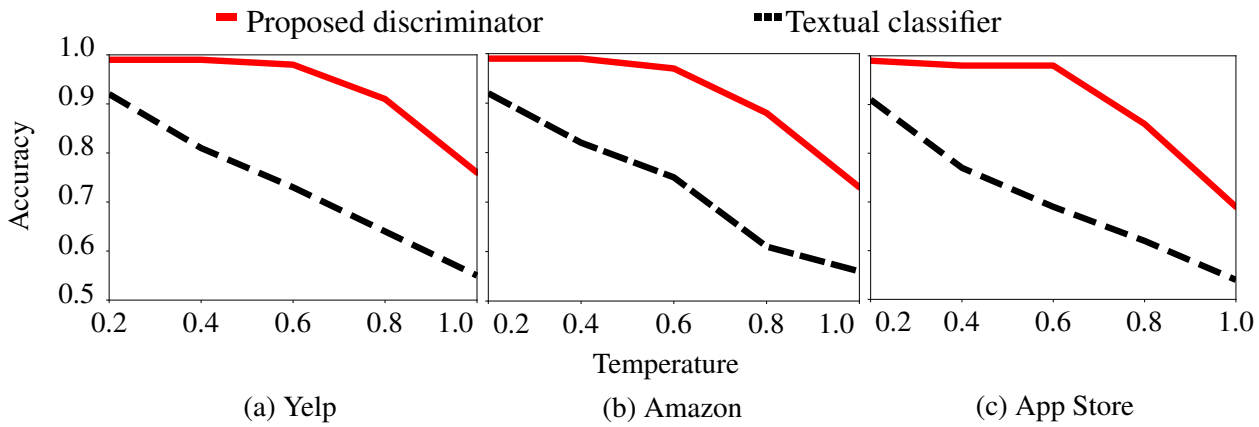
Reproducibility. To evaluate our key insight, we avoid setting up a highly tuned network and choose the hyper-parameters from singleton sets. we set the number of LSTM hidden states with respect to the median of review length in each domain (Table 4.1). we set input size, hidden size, learning rate, dropout rate, batch size, hidden layers, and optimizer to 400, 1150, 0.001, 0.1, 16, 2, and Adam respectively. we only tune the number of epochs based on the performance of the model on the validation dataset. That is, as accuracy on the validation set decreases the training process would stop to prevent the model from over-fitting.



(a) Temperature 0.4 (b) Temperature 0.8

Figure 4.10: Generated reviews tend to cluster in the embedding space. Figure best viewed in color. (Yelp)

Results. Figure 4.11 shows the detection performance across different domains at different temperatures. Due to the balanced dataset (equal representation of two classes) other evaluation metrics like recall, precision and f1-score remain similar to accuracy, so we only report the accuracy. The discriminator achieves significantly high accuracy at low temperatures [0.2, 0.4, 0.6] that is 0.99,



(a) Yelp (b) Amazon (c) App Store

Figure 4.11: The proposed discriminator detects synthetic reviews with high accuracy and performs significantly better than textual classifier.

0.99 and 0.97 on Yelp and a similar pattern is observable over other domains. However, we evaluate its performance at temperature 0.8 that is demonstrated to be the optimal temperature in the qualitative analysis. At this temperature, the discriminator is able to identify model-generated reviews with 0.92, 0.89 and 0.86 accuracy across different domains while the corresponding values obtained from the baseline textual classifier (spam detector) is 0.64, 0.62 and 0.61. These promising results motivate to consider more complex architectures as a future direction for the discriminator to detect model-generated reviews at temperature 0.8 more accurately in order to minimize the impact of automated fake reviews. In summary, model-generated reviews are detectable in the embedding space with high accuracy.

4.6 Conclusion

The DIOR framework proposed and evaluated a wide-ranging class of attacks on online review platforms based on neural language models at word-level granularity using transfer-learning. The unique attribute of this work is its domain independence, so that it can target any arbitrary review domain even with only small available review samples. The main intuition is to: (i) develop a universal model to learn general linguistic patterns in review domain and transfer this knowledge to the domain-specific language; (ii) generate high quality reviews which are competitive with real reviews and can pass the quality test by both computational-based detectors and human evaluators; (iii) demonstrate that synthetic reviews do not completely mimic the true distribution of real re-

views, so this is a powerful signal to detect automated fake reviews. The results on discriminating generated reviews are promising and model-generated fake reviews can be detected with about 90% accuracy.

5. SOPHISTICATED CLASS OF MANIPULATIONS

Investigating AI-based manipulation, we face a new challenge that motivates us to develop new generative models in order to control the topic and sentiment of generated reviews. Although DIOR shows success in generating grammatically correct, meaningful and human-readable reviews, the reviews do not necessarily reflect important topics and sentiments that users often express. It should be noted that content in review platforms are personalized for each item or service rather than the same content being duplicated across different items or services. To make the manipulation process more realistic, we validate a new approach wherein the topic and sentiment of the target are taken into account while generating the reviews. The focus of this piece is to model and evaluate topic-aware reviews.

Sequence-to-sequence architectures have been proposed to generate natural text conditioned on the characteristics defined by the first sequence in an end-to-end manner. To name a few, works in question answering [79], conversational modeling [115] and translation [80] adopt the paradigm of sequence-to-sequence architecture.

Despite this promising progress in sequence-to-sequence problem domains, such advances have not been explored in *aspect-aware review generation* primarily due to the data bottleneck. Learning a deep neural review generator requires large amounts of labeled data. While there are many existing collections of online reviews, very few have labels at the granularity of *aspects* (like price, food quality, or decor) and with *sentiment* associated with these aspects. Furthermore, it is unclear if incorporating such aspect and sentiment into a review generator would result in meaningful reviews.

In this dissertation, we explore how to make use of weak supervision to expand a small set of review segments labeled with aspects and sentiments (known as the seed set) to a large amount of unlabeled review segments demanded by data-hungry neural networks. In particular, we propose and evaluate a new framework to label and generate **a**spect **d**ependent **o**nline **re**views, named ADORE. We develop a weak labeling methodology that leverages the underlying distribution of

the reviews to infer weak (or noisy) labels. Since users express opinion on different aspects of the target in a single review, a review cannot be labeled in whole to state a specific aspect (see Figure 5.1). We propose a segmentation algorithm to split a review into its topically coherent segments and aim to label the resulting segments.

By overcoming this data bottleneck, we then show how to use these labels to train a generative model as if the ground truth labels are already available. In essence, this work aims to bridge aspect-mining and generative networks to generate product reviews conditioned on a specific aspect and sentiment. The proposed joint model encodes the aspect and sentiment that guides the review generator. Moving forward, we use aspect to refer to the combination of aspect and sentiment attributes. We employ a regularization technique into the language model to enhance its performance by giving rare words a proper probability to become visible to the generator. We also utilize an attention mechanism to reinforce the impact of the aspect encoder in predicting the next word.

We thoroughly analyze the ADORE framework using Yelp restaurant reviews to understand how effective is our weak labeling methodology, what is the optimal point for review segmentation to mitigate the adverse impact of the noise-prone nature of the labeling process, how our proposed approach performs compared to the baselines, and how much training data is required to reach a stable performance. We evaluate the quality of the generated reviews through a user-based study. We employ an ablation study to evaluate the impact of regularization technique on quality of the generated reviews. Concretely, the main contributions here are:

1. We develop a weak labeling methodology to build an aspect-aware review dataset and evaluate the effectiveness of our approach using crowd-sourced annotation.

2. We propose a joint model that learns to generate aspect-aware reviews in an end-to-end manner.

3. We evaluate the effectiveness of our proposed framework, including through user-based approaches.

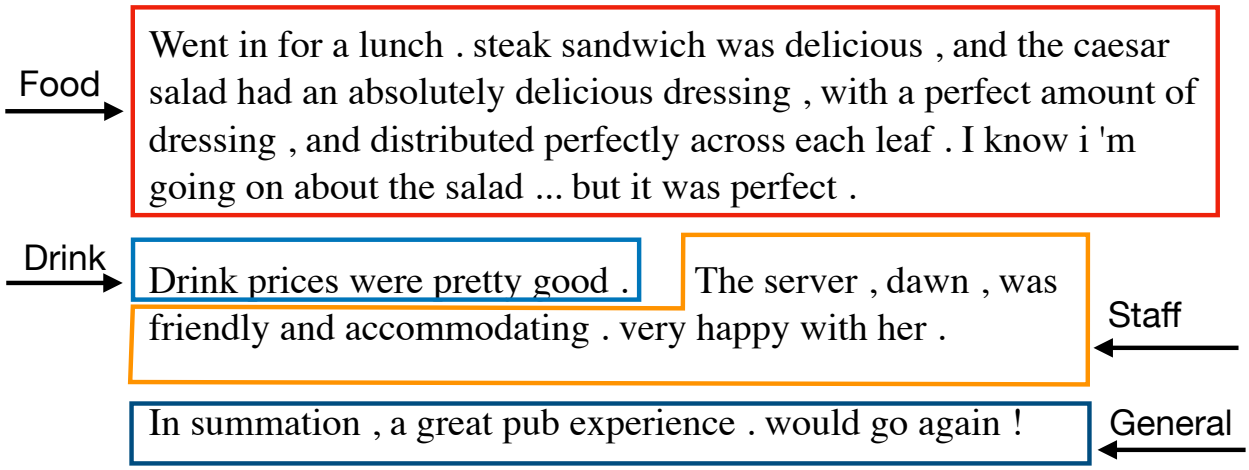


Figure 5.1: Example of Review Segmentation- A single review discusses different aspects of an item

5.1 The Proposed ADORE Framework

We propose a weak labeling methodology leveraging the underlying distribution of the reviews. Our weak labeler comprises two fundamental components: (i) *review segmentation*; and (ii) *label assignment*. Before proceeding with this approach, we pre-train a word2vec model over a large corpus of Yelp restaurant reviews (see Section 5.3.1) to obtain the word embeddings as w_i.

As illustrated in Figure 5.1, users express opinions on different aspects of the target, so a review cannot be labeled in whole to state a specific aspect. On the other hand, users typically describe an aspect in more than one individual sentence. Therefore, it is vital to detect aspect boundaries and segment reviews accordingly. In summary, the goal of review segmentation is to aggregate topically coherent *sentences* into one segment. The label assignment step aims to identify the aspect of the segments obtained from the first step using a small set of labeled data.

5.1.1 Review Segmentation

The review segmentation algorithm works at sentence level granularity and traverses through each review sentence by sentence in order to cluster coherent sentences into one segment. At its core, review segmentation is based on a sliding window technique with a window size of two. Each sentence is compared with the rightmost sentence in the previous segment. If their distance is less

than a specific threshold τ, then the sentence is added to the segment, otherwise it forms a new segment. This process continues until the end of the review.

The segmentation algorithm is based on a metric to measure the similarity between sequential sentences. We adopt the Word Mover's Distance (WMD) [116] due to its performance to measure semantic similarity between two segments of short text. Rather than relying on keyword matching, it attempts to find an optimal transformation from one sentence S to another sentence S' in the word embedding space:

$$\begin{gathered} WMD(S_i, S_j) = min \sum_{i}^{|S|} \sum_{j}^{|S'|} W_{ij} c(w_i, w_j) \\ \sum_{i}^{m} W_{ij} = 1/|S|, we \in \{1, ..., |S|\}, \sum_{i}^{n} W_{ij} = 1/|S'|, j \in \{1, ..., |S'|\} \end{gathered} \tag{5.1}$$

where $|S|$ and $|S'|$ are the length of each sentence in terms of number of words and $W_{i,j}$ is the weight of word i calculated based on a normalized bag of words (nBOW) representation of a document, so it is equal to $1/|S|$ from sentence S that is transferred to word j of sentence S'. Finally, $c(w_i, w_j)$ is the traveling cost between two words and is calculated by taking the Euclidean distance between embedding representation of words.

Algorithm 2 shows the segmentation steps for one specific review R, which can then be generalized to all the reviews in the dataset. We later show how we choose the threshold empirically.

5.1.2 Label Assignment

Now that we split reviews into coherent segments, in this step we attempt to label the segments at the aspect level. The label assignment algorithm is based on a small set of labeled data known as the seed set. Table 5.1 reports the statistics of this seed set.

The main intuition is to find semantically similar seeds to the unlabeled segments and use their labels to identify the aspect of the segments. For this purpose, we compare each sample in the seed set against the unlabeled segments using the WMD distance function. If their distance is less than a threshold τ then the segments discuss similar aspects in the semantic space and so the unlabelled segment receives the same label as the seed sample. Algorithm 3 shows the label assignment steps.

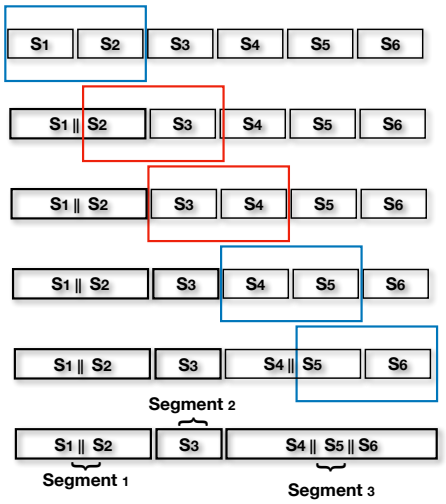


Figure 5.2: The review segmentation algorithm attempts to split a review into coherent segments by moving a sliding window over sequential sentences. Blue windows show two sentences are close in the semantic space, so they are clustered into one segment. Red windows shows the split point where two sequential sentence belong to different segments.

Algorithm 2 Review Segmentation

R : Review
S : Sentence
segments={ }
S = Split (R)
$current_{seg} = S_1$
for $i = 2$ to $|S|$ **do**
    $d = WMD(S_i, current_{seg})$
    if $d < \tau$ **then**
        $current_{seg} += S_i$
    else
        segments.add($current_{seg}$)
        $current_{seg} = S_i$
segments.add($current_{seg}$)
return segments

Table 5.1: Distribution of labels in the seed set. (+) and (-) indicate positive and negative sentiments, respectively. Amb stands for Ambience

Labels	**Food (+)**	**Food (-)**	**Amb (+)**	**Amb (-)**	**Price (+)**	**Price (-)**	**Drink (+)**	**Drink (-)**	**General (+)**	**General (-)**	**Total**
Count	446	163	102	16	19	24	28	6	407	261	1472
(%)	30.29	11.07	6.92	1.08	1.30	1.63	1.90	0.40	27.64	17.73	100

Algorithm 3 Label Assignment

seed: labeled data
seg: unlabeled segment
for $seed$ **in** $seed_{set}$ **do**
    for seg **in** $segments$ **do**
        $d = WMD(S_i, current_{seg})$
        if $d < \tau$ **then**
            $label(seg) \leftarrow label(seed)$
return label(segments)

Multi-label sentences. With respect to the review segmentation algorithm, the key assumption is that each sentence or several coherent sentences discuss only one aspect. However, expressing multiple aspects in a single sentence is a common practice when users write reviews. Table 5.2 shows some examples of such scenarios where there is no optimal point to divide it into individual parts corresponding to individual aspects. In this piece, we deal with segments with a single aspect and leave these kinds of multi-aspect segments for future work. For this purpose, we limit our experiments to the segments that receive only one label by the label assignment algorithm.

Table 5.3 demonstrates examples of segments obtained from the segmentation algorithm and their labels assigned by the label assignment algorithm across different aspects and sentiments.

5.2 Aspect-aware Review Generation

Given this bootstrapping method for overcoming the data bottleneck, we now focus on a downstream task to validate the quality of these labels. Concretely, we show in this section how to extend recent generative models of text to generate aspect-aware reviews.

The generative model consists of three main components: (i) the aspect encoder; (ii) an attention-based language model; and (iii) a regularizer to enhance the performance of the generator. In the following, we explain each of these components in turn. We use aspect to refer to the combination of both aspect and sentiment.

5.2.1 Aspect Encoder

The first key component is an aspect encoder that constructs the embedding representation for each aspect. Given an aspect, we first map them into one-hot vectors A.

Then, we employ a fully-connected gated network to encode the input aspect as $z_i = \alpha(W.A_i + b)$, where z_i is the encoder output for aspect A_i, W and b are the weight matrix and the bias vector for the linear operation respectively and are learnable parameters. $\alpha(.)$ is the non-linear activation function chosen to be ReLU. We restrict the dimension size of the aspect vectors to be identical with the generator's hidden state since they initialize the hidden states.

5.2.2 Attention-based Review Generator

Basically, neural language models [117] recurrently compute hidden states that transfer the information to the next time step. At each time step t, the network takes in the current word w_t along with the current hidden state h_t, that encodes the sequence up to the time step t, and outputs a distribution over the vocabulary for the next word. The output distribution essentially describes the probability of observing each word w' in the vocabulary given the sequence $w(<= t)$ ($P(w'|w_1, \ldots, w_t)$).

Table 5.2: Example of sentences with more than one aspect.

Multi-label Review Sentences	**Labels**
food is delicious only down fall is price and portion size .	Food (+) Price (-)
prices are reasonable and the food tastes great .	Price (+) Food (+)
the food was excellent and we highly recommend the business .	Food (+) General (+)
beer was all good and food generous and tasty .	Drink (+) Food (+)
highly recommend this location for quality service and price .	General (+) Price (+)

Table 5.3: Example of segments and their corresponding labels obtained from the segmentation and label assignment algorithms respectively across different aspects and sentiments. (+) and (-) indicate positive and negative sentiments, respectively.

Aspect-specific Review Segments	**Label**
steak sandwich was delicious and the caesar salad had an absolutely delicious dressing with a perfect amount of dressing and distributed perfectly across each leaf . we know we m going on about the salad .	**Food (+)**
today was my second visit to the place after having a good first experience but we am so disappointed with the quality of the food that we can say it has been my worst experience of food in months the sun dried tomatoes very absolutely stale to an extent that they tasted bitter the pizza base was so thick that it was uncooked and soggy the four cheese blend tasted completely different than the last time and so did the pesto sauce . no consistency with food quality .	**Food (-)**
the ambiance is nice too . it s a bit dark but they have this nice light display above on the ceiling made with mason jars . there is a comfy seating area in the bar area that s nice too .	**Ambience (+)**
however the one thing that surprised me was how dirty the restroom was in this restaurant . the floor was really dirty and toilet papers were unwell kept . the restaurant could at least have someone maintained the restroom in good shape and clean because this will reflect on how one maintains the cleanliness of the place .	**Ambience (-)**
the price is very reasonable for a family of four with plenty of leftovers to take home .	**Price (+)**
my wife we had a groupon for this place and for the price it was very poor value quality .	**Price (-)**
we had a nice glass of california cabernet . the wine list while not expansive was good . the bartender we had seemed to have a nice knowledge of what was going on with the wine that encompassed it .	**Drink (+)**
we ordered a glass of Merlot that was delivered to me in a dirty glass . the waitress was very polite and went to get me a new glass of wine but we was still unimpressed at that point .	**Drink (-)**
highly recommend for lunch . even during lunch rush it was not super packed . this would be a good place for a lunch meeting .	**General (+)**
we am not sure why anyone would like this place . the only thing it has going is location and that is simply not enough not for me .	**General (-)**

With this definition in mind and given the output of the aspect encoder z, our generative model learns to produce reviews based on information in z in addition to the information encoded in the hidden states h_t. For this purpose, the encoder output z initializes the first hidden states and in order to reinforce its impact throughout the network, an attention layer is introduced to capture the soft alignments between z and h_t.

Each aspect-specific review sample is a row in the input review matrix R of shape $n \times T$ where n is the number of samples in the training data and T is the number of time steps in the recurrent neural network (also interpreted as the size of back propagation through time). Since the generator aims to predict the next word at each time step, the output matrix (Y) is just like the input matrix but shifted by one word to the right defined as $Y = R[1:]$. Without loss of generality, we now describe the architecture of the review generator at time step t. It encompasses four main layers: Embedding, stacked-GRUs, Attention, and Decoder.

The Embedding layer is a trainable matrix W_e that learns the low-dimensional representation of the input tokens. The matrix shape is defined by vocabulary size V and embedding size. We then employ a L-layer GRU recurrent network to capture the dependency among review words:

$$h_t^1 = GRU(w_t, h_{t-1}^1)$$

$$\dots$$

$$h_t^L = GRU(h_t^{L-1}, h_{t-1}^L)$$

where h_t^i is the hidden state calculated by i^{th} layer for word w_t. It should be noted that h_0^i are initialized with aspect encoder output ($h_0^1, \dots, h_0^L = z$). In addition, the attention layer incorporates the aspect information into the hidden state calculated at the last layer for word w_t. In particular, given the hidden state h_t^L and aspect z vectors, we first apply a linear transformation to obtain a score for each vector as $s_t = (h_t^L || z) \times W_s$, where W_s is a learnable parameter vector of shape 1 $\times$ hidden size and the concatenation of two vectors (h_t^L, z) gives a matrix of shape 2 $\times$ hidden size. The s_t determines the score for each vector. Then the attentive weight is calculated with a *softmax*

function over s_t values:

$$a_t^v = \frac{exp(s_t^v)}{\sum_v exp(s_t^v)}$$

where v could be either a hidden state or aspect vector. a_t^v is the weight indicating the relatedness between aspect information around the next review word w_{t+1} to be predicted. Therefore, we update the hidden state in the last layer as $h_t^L = a_t^{h_t^L}.h_t^L + a_t^z.z$.

The updated h_t^L is fed into the Decoder layer. The Decoder layer is a linear transformation that decodes the hidden state to predict the next word as $o_t = W_o.h_t^L + b_o$, where W_o and b_o are the weight matrix and the bias vector for the linear operation respectively and are learnable parameters. The output vector o_t is then activated by the *softmax* function to produce the probability distribution over the words in the vocabulary $\hat{y}_t = softmax(o_t)$.

The output $\hat{y}_t$ is then compared with the ground truth y_t through cross-entropy loss as:

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{|V|} y_{t,j} \log \hat{y}_{t,j} \tag{5.2}$$

Both y and $\hat{y}$ are V-dimensional vectors, where V is the size of vocabulary and T denotes the length of the sequence or number of the time steps in the RNN. The network parameters are then updated over multiple iterations to minimize the loss value.

5.2.3 Diversity-enforcing Review Generator

Neural language models are often biased towards frequent patterns in training data while ignoring rare words. Specifically, the size of the vocabulary in the review domain is small and the generator tends to generate repeated patterns. Therefore, we employ the diversity-enforcing penalty function proposed in [118] as a regularizer to the loss function. In particular, it adds the aggregate cosine similarity between every pair of words in the vocabulary to the $J(\theta)$.

$$R = \frac{1}{|V|^2}\sum_{i}^{|V|}\sum_{j\neq i}^{|V|} \frac{w_i^T w_j}{\|w_i\|\|w_j\|} \tag{5.3}$$

Table 5.4: Examples of generated reviews conditioned on input label.

Input Label	Aspect-aware Generated Reviews
Food (+)	The meat is amazing and the portions are perfectly balanced.
Food (-)	The beef was dry and the chicken was tough.
General (+)	The staff is impeccable and the food is exceptional and it's not a huge thing for the money
General (-)	we would say they were extremely rude to us.
Amb (+)	Great dining experience with a great vibe.
Amb (-)	The room was nice, but the only thing we had was the lighting in the bathroom.
Price (+)	The prices are reasonable and the service was very good.
Price (-)	The price is a little expensive, but we would expect going for $ 20 for a steak.
Drink (+)	That was a perfect drink spot to go with a pitcher of champagne on the menu.
Drink (-)	They have a full bar with a decent selection of beers on tap, there were no descriptions of the beers on tap, but we 'm sure that's in a pint glass.

Intuitively, when training the model with cross-entropy loss, at time step t the embedding of the corresponding word in the ground truth w_{t+1} is pushed to become close to the output vector o_t in order to get a larger likelihood, while the embeddings of the other words in the vocabulary are pushed towards the negative direction of o_t to receive a smaller likelihood. According to Zipf's law the word frequency in the training data is very low compared to the size of the corpus. For a concrete example, in our dataset the frequency of the popular word "is" is only about 1.22% while frequency of unpopular words drops drastically. Therefore, the embedding of the rare words are pushed towards negative directions of most output vectors and they get low probability in the final probability distribution $\hat{y}$. To encourage these rare words to become visible to the predictor, we need a mechanism to push their embedding toward a positive direction correlated with output vectors.

The idea of the regularization term is to increase the angle between words, i.e., minimizing the cosine similarity between any pair of word embeddings, so they are not pushed in one direction. We later in Section 5.3.2 show the effectiveness of the diversity-enforcing regularizer in producing

diverse aspect-specific reviews.

Table 5.4 reports some samples of *generated* aspect-aware reviews across different aspects and sentiments.

5.3 Experiments

We first describe the dataset. Then, we evaluate our proposed labeling methodology qualitatively and quantitatively. We also evaluate the quality of generated aspect-aware reviews from different angles.

5.3.1 Data

Seed data. This dataset originally was introduced in the *SemEval* challenge [119] in 2016 for *Aspect Based Sentiment Analysis* task. It contains 1,472 pairs of reviews and their labels after removing duplicate samples. Moving forward, we use label to refer ro the aspect and sentiment of the review. The dataset represents five different aspects as *Food, Drink, Price, Ambience and General* coupled with the two sentiments as positive and negative (10 labels in total).

Yelp Dataset. This dataset was released as part of round 13 of the Yelp challenge in January 2019: *https://www.yelp.com/dataset/challenge*. We use this dataset as the input to ADORE consisting of segmentation and labeling methods and further it is used as the ground truth required by the generative model. It has about 4M reviews for about 60k restaurants with 522M tokens. We picked the top 51k most frequent words to build our vocabulary and keep only reviews with words from the vocabulary, ending up with 2,791,379 ($\sim$3M) reviews consisting of 258M tokens.

5.3.2 Experimental Design

Review pre-processing: The segmentation algorithm produces $\sim$7M segments out of the initial $\sim$3M reviews. However, not all of the segments are subjective. For example, the sentence *"We went there for lunch."* does not express any opinion. To address this issue, we filter out subjective segments using the TextBlob library $https://planspace.org/20150607-textblob\_sentiment/$ that calculates polarity of the text. It outputs about $\sim$3.5M reviews to feed to the weak labeler.

Hyperparameters. We are interested in a general model that performs robustly across different

labels, so we avoid setting up a highly tuned network and choose the hyper-parameters from singleton sets. We set the number of time steps in the recurrent model with respect to the maximum review length, i.e., 70. We set input size, hidden size, learning rate, dropout rate, batch size, number of GRU layers, and optimizer to 100, 100, 0.001, 0.2, 20, 2, and Adam respectively. We then adjust the learning rate decayed by 10 every 5 epochs. We only tune the number of epochs based on the performance of the model on the validation dataset. We empirically find that the model reaches a stable performance after about 10 epochs. We also split the review samples into training, testing and validation sets with ratio of 90%, 5% and 5% respectively.

Review Generation. The model takes in the desired aspect and the starting word w_0 which we define by a special token as $< sor >$ in training samples. At each time step t, it takes in the word predicted at the previous time step (w_{t-1}), the hidden state h_{t-1} and aspect vector to predict the distribution for the next word w_t and also updates the hidden state to h_t. By feeding w_t back to the model, it produces another probability distribution to predict the next word. The generation process continues until the model predicts the end of the review identified by a special token as $< eor >$.

5.3.3 Experimental Results and Discussion

We evaluate the ADORE framework from two dimensions: the effectiveness of the weak labeling methodology and the performance of the generative model.

5.3.3.1 Evaluation of Labeling Methodology

We first determine the optimal threshold for segmentation. Then we compare our segmentation algorithm with existing baselines. Finally, we conduct a user study to assess the quality of labels. It should be noted that labeling evaluation is heavily based on a user study as the end goal is to evaluate how the quality of segments and labels are perceived by end users.

Determining the distance threshold (τ) for segmentation algorithm. We conduct a user study to identify the optimal point for segmentation. This value determines whether to merge two sequential sentences into one segment or split them into different segments. We empirically observe the

distance between two sequential sentences varies from 1.00 to 1.35. Note that WMD is based on Euclidean distance over normalized embedding vectors so the distance value varies from 0 – two exact sentences – to the maximum value of 2. We pick 105 sample reviews and deploy the segmentation algorithm on each review using different threshold values. We ask seven expert labelers to identify the threshold that closely replicates the segmentation that would be done by human segmenters. Table 5.5 shows the distribution of the thresholds determined by human judges indicating value of 1.1 gives the more accurate segmentation results.

Comparison with Baselines. We study the performance of our proposed segmentation algorithm with three baselines.

Sentence-level segmentation: This method breaks down a review into its sentences and takes each sentence as one segment with the assumption that an aspect of the target is discussed in a single sentence.

Review-level segmentation: Alternatively, we consider the whole review as a single segment with this assumption that each individual review discusses only one aspect of the target.

Text segmentation based on semantic word embeddings [120]: This algorithm is based on word embeddings for text segmentation and demonstrates state-of-the-art performance for an unsupervised method and follows the similar scenario as our work. Briefly, it assumes the whole text to be one segment and divides the text into multiple segments with sentence level granularity during a recursive process. Intuitively, a segment is coherent if the similarity between its words w_i and the segment v as a whole is maximized.

However, this approach is designed to find the topic boundaries in long text such as scholarly articles. In this study, we aim to show that our proposed unsupervised segmentation algorithm outperforms the baselines when it comes to splitting a review (which is typically much shorter than a scholarly article) into coherent aspect-specific pieces. Table 5.6 shows a motivating example of different segmentation baselines.

There is a standard text segmentation evaluation metric known as P_k. It measures the error, so lower scores indicate higher accuracy. However, it is based on a ground-truth of segments that

Table 5.5: Human labelers identify 1.0, 1.1 and 1.2 as optimal thresholds for review segmentation.

Threshold (τ)	1.0	1.1	1.2	1.25	1.3	1.35
number of votes	25	31	27	12	9	1

are not readily available in our scenario. For this purpose, we sample 100 reviews and manually split them into their segments and aim to recognize the segmentation algorithm that produces segments most closely to the segments obtained from expert evaluators. We refer to the segments belonging to one review in the ground-truth as *reference partition* and segments obtained from the segmentation algorithm as *hypothesized partition*.

P_k [121] is the probability of segmentation error. It takes a window of fixed size k, and moves it across the review. At each step, it examines whether the hypothesized partition is correct based on the separation of the two ends of the window. In particular, it counts the number of disagreements between the reference and hypothesized partitions. P_k is defined as:

$$P_k = \frac{1}{N-k} \sum_{i=1}^{N-k} [\sigma_{hyp}(i, i+k) \neq \sigma_{ref}(i, i+k)]$$

Where $\sigma_{ref}(i, j)$ is a binary function whose value is one if the sentences we and j belong to the same segment and zero otherwise according to ground-truth. Similarly, $\sigma_{hyp}(i, j)$ is one if sentences we and j exist in the same segment and zero otherwise according to the segmentation algorithm. In the review domain due to short texts and sentence-level granularity for segmentation, we set the window size (k) to 1. N refers to the number of sentences in the review.

We calculate the P_k for each review segmented automatically by the algorithm against the ground-truth and take the average over all reviews. Table 5.7 shows the performance across different segmentation algorithms. ADORE produces P_k score with a lower value compared to its alternatives indicating its effectiveness in finding the aspect boundary inside reviews.

How does automated labeling perform compared to manual labeling? Here, we are interested in investigating if our weak labeler is comparable with manual labeling. For this purpose, we set up

Table 5.6: An review sample is segmented at review-level and sentence-level and by state of the art text segmentation algorithm, our proposed ADORE algorithm and human expert labelers. ADORE segmentation is the closest to replicating human segmentation.

Segmentation Method	Segments
Review-level (one segment)	dinner was fantastic . service was great we started with the corn soup and the tuna tartare . we shared the filet and scallops . both delicious entrees . we did not realize the steak came with potatoes and ordered two sides mac and cheese and shishito peppers . we thought the peppers were really hot but we m a whimp we guess . we will definitely come back .
Sentence-level (seven segments)	dinner was fantastic .
	service was great we started with the corn soup and the tuna tartare .
	we shared the filet and scallops .
	both delicious entrees .
	we did not realize the steak came with potatoes and ordered two sides mac and cheese and Shishito peppers .
	we thought the peppers were really hot but we m a whimp we guess .
	we will definitely come back .
Text Segmentation [120] (six segments)	dinner was fantastic .
	service was great we started with the corn soup and the tuna tartare .
	we shared the filet and scallops . both delicious entrees .
	we didn t realize the steak came with potatoes and ordered two sides mac and cheese and shishito peppers .
	we thought the peppers were really hot but we m a whimp we guess .
	we will definitely come back .
ADORE (Three segments)	dinner was fantastic . service was great we started with the corn soup and the tuna tartare . we shared the filet and scallops . both delicious entrees .
	we did not realize the steak came with potatoes and ordered two sides mac and cheese and shishito peppers . we thought the peppers were really hot but we m a whimp we guess .
	we will definitely come back .
Ground-truth (Three segments)	dinner was fantastic . service was great we started with the corn soup and the tuna tartare . we shared the filet and scallops . both delicious entrees .
	we did not realize the steak came with potatoes and ordered two sides mac and cheese and shishito peppers . we thought the peppers were really hot but we m a whimp we guess .
	we will definitely come back .

Table 5.7: ADORE outperforms the baselines by segmenting the reviews into their coherent topics more accurately.

	Sentence-level	Review-level	[120]	**ADORE**
P_k	0.281	0.478	0.283	**0.265**

Table 5.8: Majority of the automated labels are recognized as accurate by human evaluators with > 80% acc.

Label	**Accuracy**	**Label**	**Accuracy**
Food (+)	94.33	Food (-)	85.66
General (+)	87.66	General (-)	81.00
Ambience (+)	81.33	Ambience (-)	77.66
Price (+)	86.00	Price (-)	68.00
Drink (+)	82.23	Drink (-)	57.66

a crowd-based user study to verify if the labels are assigned truthfully according to human readers. We post 100 surveys on Amazon Mechanical Turk (AMT) each including a guideline and a set of reviews for which we seek a label from Turkers. The guideline has two major points: (i) it shows a sample of reviews along with their labels from the seed set to provide a context on how reviews and labels are paired with each other, (ii) it asks Turkers to label the reviews through a series of multi-choice questions. We design 100 surveys each with 10 reviews to cover all the labels. Each unique survey is assigned to three workers, i.e., 3 HITs (Human Intelligence Task) per task, giving us a total of 300 surveys and 3,000 questions.

To ensure the quality of responses, we insert a trivial question into each survey, which asks the Turker to check if a mathematical equation is False or True. It helps to manage the risk of blindly answered surveys. Furthermore, we only accept surveys from Turkers with approval rating of at least 95% and those who dwell on the survey for at least 7 minutes. We also restrict our tasks to workers located in the United States to guarantee English literacy.

Table 5.8 demonstrates the performance of the ADORE labeling process against human judg-

ment across various labels. The key point is that the majority of the labels are found accurate by human evaluators with at least 80% accuracy. However, the accuracy for labels Price/negative and Drink/negative is relatively low and we can relate this to the fact that these labels do not have a significant representation in the seed set (Table 5.1). We aim to release the annotated reviews to contribute to the research community.

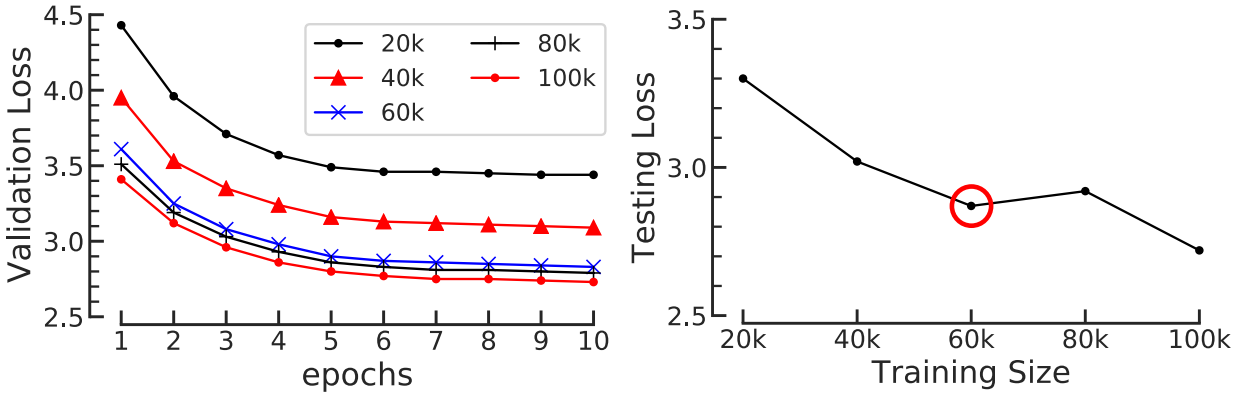
5.3.3.2 Evaluation of Generative Model

We evaluate the proposed joint generative model in generating aspect-aware reviews from three dimensions. We first test how much labeled data is required to reach a stable performance. Second, we evaluate how end users perceive the quality of aspect-aware reviews. Finally, we do an ablation study on the impact of diversity-enforcing regularizer. Note that our aim complements efforts to improve language models and we focus on automated labeling at the aspect-level and propose to generate high-quality aspect-aware reviews as the downstream task for the labeled data.

How much labeled data is required to reach a stable performance? Although we propose to build a ground truth by an automated approach, it is critical to understand how much data is required to feed neural networks. In particular, when the target review domain for a specific label is not sufficiently large we aim to discover how much labeled data would address the data scarcity problem. For example, we observe that the distribution of the labels obtained from the weak labeling process not only is not uniform but extremely biased towards popular aspects like food. For instance,
we receive $\sim$300k samples labeled as food/positive while only 412 samples are labeled as drink/negative.

For this purpose, we plot the validation loss during training epochs with various training size in Figure 5.3a. We gradually increase the training size starting with 20k reviews. This shows that approximately 60k samples are required to improve the performance at its finest. Figure 5.3b plots testing loss versus the training size and echoes the same intuition.

How are aspect-aware generated reviews perceived by end users? Regardless of the model performance with respect to labeling confidence and training size, the real test is to evaluate how



(a) Validation loss in training

(b) Testing loss w.r.t data size

Figure 5.3: Stable performance with at least 60k samples.

Table 5.9: Majority of the generated aspect-aware reviews are perceived as reliable by human evaluators with 90% acc.

Label	**Accuracy**	**Label**	**Accuracy**
Food (+)	93.07	Food (-)	97.69
General (+)	86.15	General (-)	85.38
Ambience (+)	97.69	Ambience (-)	90.76
Price (+)	90.00	Price (-)	83.07
Drink (+)	90.76	Drink (-)	33.84

reviews are perceived by human readers. Similar to label assessment, we launch a crowd-based user study by posting surveys on AMT. We follow similar guidelines to those mentioned in the previous section to ensure the quality of the answers.

We design 100 surveys each with 10 generated reviews at various aspects. We assign 3 HITs per task, giving us a total of 300 surveys and 3,000 questions. We ask Turkers to label the model-generated reviews through a multi-choice questions based on the aspect.

From Table 5.9, we observe that generated reviews stay with the desired aspect with higher than 90% accuracy for a majority of the labels. For example, 93% and 97% of reviews on Food (positive and negative) are perceived equally by the model and the human evaluators while this number is

Table 5.10: The regularized model generates more diverse reviews. TTR (Token Ratio), MTLD (Textual Lexical Diversity)

	Reg	Food (+)	Food (-)	Amb (+)	Amb (-)	Price (+)	Price (-)	Drink (+)	Drink (-)	Gen (+)	Gen (-)
Word count	✓	996	1126	1122	1422	575	995	998	1600	888	1351
	×	986	1096	1090	1289	601	1136	951	1431	983	1424
Unique terms	✓	**186**	**174**	**162**	**306**	45	**168**	**162**	**244**	149	**249**
	×	165	162	130	247	**51**	164	139	202	**175**	253
TTR	✓	**0.18**	**0.15**	**0.14**	**0.21**	0.08	**0.16**	**0.16**	**0.15**	0.16	**0.18**
	×	0.16	0.14	0.11	0.19	0.08	0.14	0.14	0.14	**0.17**	0.17
MTLD	✓	**65**	60	**65**	**149**	**28**	**85**	**79**	**117**	77	**150**
	×	55	60	46	122	24	84	71	96	**95**	137

34% for drink/negative. We can relate this to the fact that the label *Food* has a better representation in both seed set Table 5.1 and our expanded dataset as it is the main topic of discussion when writing a review for a restaurant.

How does the regularized language model improve diversity? Here, we are interested to evaluate the impact of regularization on the quality of the generated reviews in terms of their diverse vocabulary. To do this, we develop the model in the presence and the absence of the optimization term described in 5.2.3. The testing loss is 2.80 and 2.93 respectively indicating that regularized model adapt with significantly more diverse patterns. However, we take one step further and evaluate this feature qualitatively as well.

We pick a sample of 1k reviews generated by each of the models (2k in total) across various labels. We then concatenate the samples in the same label category and report the number of words w, the number of unique terms t, and the ratio of the tokens (TTR) calculated as t/w in Table 5.10. The main intuition is that the more diverse reviews have a larger number of unique terms and higher values for TTR. However, in the review domain, the number of unique words are limited and TTR falls as we add more samples. Therefore, the MTLD (Textual Lexical Diversity) metric is proposed [122] that counts the number of words before TTR falls below a given threshold. By setting the threshold to 0.5, we see from Table 5.10 that MTLD for the regularized model is higher

indicating they have more diverse vocabulary such that the more number of words are needed to meet the TTR threshold. We also report the diversity metrics for aggregation of the reviews across all the labels.

5.4 Conclusion

Investigating AI-based manipulation, we face a new challenge that motivates us to develop new generative models in order to control the topic and sentiment of the generated reviews. Although DIOR shows success in generating grammatically correct, meaningful and human-readable reviews, it does not reflect topic and sentiment of the generated reviews. In this dissertation chapter, we explored how to make use of weak labels in order to generate aspect-specific online reviews when sufficient data required by neural networks are not available. The main intuition is to: (i) build a ground truth of aspect-specific reviews automatically, (ii) propose a generative model that produces reviews conditioned on input aspects; and (iii) evaluate the quality of the labels and generated reviews.

6. CONCLUSION AND FUTURE WORK

This dissertation investigated manipulation powered by humans and AI in large-scale review platforms and proposed several frameworks with inter-related techniques to combat them. In this chapter, we first give a summary of our contributions and findings, and then provide multiple open questions for future research.

6.1 Summary of Contributions and Findings

Crowd-based review manipulation. In Chapter 3, we described crowd-based manipulation from sampling manipulation traces to characterizing and detecting manipulators and targets of manipulation attacks. In particular, we developed a cross-domain data collection approach to obtain samples of crowd manipulation given the previous efforts miss a ground truth for evaluation. We proposed the TwoFace framework for detecting manipulators. The important findings are that fraudulent users engage in a mix of legitimate and deceptive behaviors making this a challenge to identify them through behavioral characteristics. Also, crowd campaigns do not form a dense block over the co-review graph making this a challenge for dense-block detection algorithms. Therefore, the proposed framework is built upon *neighborhood-based characteristics*. The important finding is that although individual behaviors may be easy to mask, the collective campaign organization can be uncovered by exploiting the network around reviewers and products. However, propagation of suspiciousness misses fraudulent users that are distant in the suspiciousness graph. TwoFace is enhanced by an embedding-based approach that maps users into a low-dimensional embedding space and captures community structure of the graph. The main finding is that TwoFace uncovers many manipulators even when the system is seeded with only a few fraudulent samples with 93% recall and 77% precision, outperforming a state-of-the-art baseline. Although crowd campaigns do not show lockstep behavior that is the focus of dense-block detection algorithms, our embedding-based TwoFace framework is able to detect them with high accuracy.

While fake review writers and fake reviews serve as a building block of an attack, the ultimate

goal is often to manipulate a specific *target*. Knowing which products (or services,etc.) are targets of an attack, we can defend the target from ongoing threats, develop more robust defensive countermeasures for future threats and develop robust recommendation techniques to diminish the impact of fake reviews on recommendation results. This dissertation built a ground truth of products that have been targeted by crowd manipulation and it gave us this opportunity to study manipulation at product level unlike previous works that focus on detection of either users or reviews. Therefore, we proposed the TOmCAT framework for detecting targets of manipulation. It first identifies patterns of crowd-based attacks as promotion, wherein a crowd seeks to manipulate the product rating of a new product; and restoration, wherein a crowd seeks to counteract a low rating from a legitimate reviewer. We found that the crowd campaigns are relatively small (soliciting between 5 to 10 reviews) that suggests they do not leave obvious patterns of synchronized behavior that could aid in their detection. Another important observation is that crowd reviews demonstrate high-quality content making existing methods that rely on signals of poor review quality to be ill-suited to uncover them. Based on these observations and identified manipulation patterns, TOmCAT formulated crowd attack foot-prints into a suite of features based only on timing and sequence of product ratings and proposed a neural-based model to learn the review behavior at product level. It outperformed several baselines and can be generalized over other review platforms. However, the important finding is that although TOmCAT can uncover target products with high accuracy by addressing existing attacks, strategic attackers can potentially create hard-to-detect behavioral patterns by undermining timing-based footprints. we proposed TOmCATSeq to model review behavior in an end-to-end way where it can restore the detection performance against strategic attackers to some extent (Figure 3.18).

AI-based review manipulation. In Chapter 4, we described AI-based manipulation from different angles as generation, validation, and detection. In particular, we proposed a universal model capable of generating synthetic reviews across different review domains such as Yelp, Amazon and App store. The main finding is that the traditional linguistic-based spam detector may not distinguish synthetic reviews from real reviews. This dissertation also found that the majority of

synthetic reviews (80%) are recognized as real by humans indicating the power of the proposed model in generating realistic reviews. The results of a user study show that human readers are more sensitive to repetition errors than they are to small grammar mistakes. To validate the important question of whether neural language models are capable enough to take the place of crowd campaigns, this dissertation launched a user study to evaluate how end users perceive the synthetic reviews in comparison with fake reviews written by crowd workers and found that users find reviews generated by the proposed DIOR as reliable as fake reviews written by manipulation campaigns. The proposed DIOR beats the state-of-the-art model in generating effective reviews and its unique feature is being domain independent and can target any arbitrary review domain even with small available review samples. We found transfer learning not only facilitates the domain shift but also improves the performance. In another examination, this dissertation showed that transferred models need lower number of training samples compared to universal model to reach the stable performance. Although model-generated reviews can pass the human and linguistic-based test, they have different underlying distributions compared to human-written reviews, so they can be detected with high accuracy based on their embedding representation.

Sophisticated AI-based manipulation. In Chapter 5, we described a new class of manipulations that can control the topic and sentiment of the generated reviews. The main challenge is to develop a dataset of reviews at topic and sentiment level. This dissertation developed a weak labeling methodology to build an aspect-aware review dataset and evaluated the effectiveness of the labels using crowd-sourced annotation. It also showed that the proposed segmentation algorithm to split a single review into its topically-coherent sub-reviews outperforms existing methods. As part of the ADORE framework, we developed a joint generative model that generates aspect-aware reviews. End users perceived aspect-aware generated reviews as realistic with 90% accuracy.

6.2 Future Work

Explainable review manipulation patterns. This dissertation proposed several frameworks to detect online manipulations enabled by different actors from crowd to AI-based models. One open question is given a deep learning model with a high detection rate of manipulation, how can

we explain the output of the detection model? In other words, what are those patterns that guide the model to distinguish between normal and anomalous behaviors? As a future direction, we can explore the explainability aspects of these models. This helps us understand what features in the data the neural model is learning and whether the resulting model is biased towards a particular aspect of the data. Once we understand the relations between data and model, we can i) refine the initial detection model for better performance; and ii) we can further improve the model to address situations where attackers exploit the bias in the model to remain undetected.

Explainable review manipulation detection models. The explainability of end-to-end models for manipulation detection is under-studied for a couple of reasons. i) Due to the lack of labeled data for manipulated contents, the detectors cannot benefit from supervised learning that is the foundation of deep learning techniques for classification tasks. ii) The existing techniques for explainable fake news detection cannot be directly applied to explain fake behaviors in general since they come from different underlying data sources. News is fact-based and can be validated in accordance with existing facts while user behaviors like promoting a *hashtag* through fake retweet or promoting an item through fake reviews are subjective actions. Reviews are opinion-based meaning different people may have completely different opinions towards the same target so as we showed in this dissertation we need contextual information beyond review content to identify patterns of manipulation. Since we curated a dataset of review manipulation wherein the existence of fake reviews is evident, we are able to validate the proposed explainability techniques and expand our experiments to alternative datasets with respect to generalization.

Extending the work on other review platforms beyond e-commerce systems. We have performed the evaluation of the proposed frameworks over e-commerce review datasets throughout this dissertation. Investigating manipulation on other platforms where enable users to leave comments such as FCC can be considered as a future research direction given the intentions and interactions are different from e-commerce platforms.

How to measure impact of fake reviews on consumers. Assuming the intention of the fake reviews in a site like Amazon is to promote the items' ratings in order to increase the revenue. An

open question is how we can measure the impact of fake reviews on revenue or to what degree review manipulation is successful to reach this goal?

REFERENCES

[1] R. Murphy, "Local consumer review survey 2017, https://www.brightlocal.com/research/local-consumer-review-survey-2017/, last access: 20/09/2020," 2017.

[2] M. Luca and G. Zervas, "Fake it till you make it: Reputation, competition, and yelp review fraud," *Management Science*, 2016.

[3] J. Kao, "More than a million pro-repeal net neutrality comments were likely faked, https://tinyurl.com/fcc-nn, last access: 02/26/2018," 2017.

[4] Y. Yao, B. Viswanath, J. Cryan, H. Zheng, and B. Y. Zhao, "Automated crowdturfing attacks and defenses in online review systems," in *CCS*, ACM, 2017.

[5] S. Kumar and N. Shah, "False information on web and social media: A survey," *arXiv preprint arXiv:1804.08559*, 2018.

[6] P. Kaghazgaran, J. Caverlee, and M. Alfifi, "Behavioral analysis of review fraud: Linking malicious crowdsourcing to amazon and beyond.," in *ICWSM*, 2017.

[7] P. Kaghazgaran, J. Caverlee, and A. Squicciarini, "Combating crowdsourced review manipulators: A neighborhood-based approach," in *WSDM*, 2018.

[8] P. Kaghazgaran, M. Alfifi, and J. Caverlee, "Tomcat: Target-oriented crowd review attacks and countermeasures," in *ICWSM*, 2019.

[9] P. Kaghazgaran, M. Alfifi, and J. Caverlee, "Wide-ranging review manipulation attacks: Model, empirical study, and countermeasures," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 981–990, 2019.

[10] P. Kaghazgaran, J. Wang, R. Huang, and J. Caverlee, "Adore: Aspect dependent online review labeling for review generation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1021–1030, 2020.

[11] P. Kaghazgaran and J. Caverlee, "Towards an automated writing assistant foronline reviews," in *CHI AutomationXP20 Workshop*, 2020.

[12] A. Zubiaga, M. Liakata, R. Procter, G. Wong Sak Hoi, and P. Tolmie, "Analysing how people orient to and spread rumours in social media by looking at conversational threads," *PloS one*, 2016.

[13] L. Zeng, K. Starbird, and E. S. Spiro, "Rumors at the speed of light? modeling the rate of rumor transmission during crisis," in *HICSS*, IEEE, 2016.

[14] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science*, 2018.

[15] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, "Copycatch: stopping group attacks by spotting lockstep behavior in social networks," in *WWW*, 2013.

[16] C. Silverman, "Lies, damn lies, and viral content: How news websites spread (and debunk) online rumors, unverified claims and misinformation," *Tow Center for Digital Journalism*, 2015.

[17] E. Tacchini, G. Ballarin, M. L. Della Vedova, S. Moret, and L. de Alfaro, "Some like it hoax: Automated fake news detection in social networks," *arXiv preprint arXiv:1704.07506*, 2017.

[18] S. Kumar, R. West, and J. Leskovec, "Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes," in *WWW*, 2016.

[19] N. Jindal and B. Liu, "Opinion spam and analysis," in *WSDM*, ACM, 2008.

[20] F. H. Li, M. Huang, Y. Yang, and X. Zhu, "Learning to identify review spam," in *Twenty-second international joint conference on artificial intelligence*, 2011.

[21] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, "Finding deceptive opinion spam by any stretch of the imagination," *arXiv preprint arXiv:1107.4557*, 2011.

[22] H. Li, Z. Chen, A. Mukherjee, B. Liu, and J. Shao, "Analyzing and detecting opinion spam on a large-scale dataset via temporal and spatial patterns.," in *ICWSM*, 2015.

[23] G. Wang, S. Xie, B. Liu, and S. Y. Philip, "Review graph based online store review spammer detection," in *ICDM*, IEEE, 2011.

[24] L. Akoglu, R. Chandy, and C. Faloutsos, "Opinion fraud detection in online reviews by network effects.," *ICWSM*, 2013.

[25] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian, "Rev2: Fraudulent user prediction in rating platforms," in *WSDM*, 2018.

[26] S. Xie, G. Wang, S. Lin, and P. S. Yu, "Review spam detection via temporal pattern discovery," in *KDD*, 2012.

[27] J. Ye, S. Kumar, and L. Akoglu, "Temporal opinion spam detection by multivariate indicative signals," *arXiv preprint arXiv:1603.01929*, 2016.

[28] B. Hooi, N. Shah, A. Beutel, S. Günnemann, L. Akoglu, M. Kumar, D. Makhija, and C. Faloutsos, "Birdnest: Bayesian inference for ratings-fraud detection," in *SIAM (SDM)*, SIAM, 2016.

[29] A. Mukherjee, V. Venkataraman, B. Liu, and N. S. Glance, "What yelp fake review filter might be doing?," in *ICWSM*, AAAI, 2013.

[30] S. Rayana and L. Akoglu, "Collective opinion spam detection: Bridging review networks and metadata," in *KDD*, ACM, 2015.

[31] A. Mukherjee, A. Kumar, B. Liu, J. Wang, M. Hsu, M. Castellanos, and R. Ghosh, "Spotting opinion spammers using behavioral footprints," in *KDD*, ACM, 2013.

[32] J. Ye and L. Akoglu, "Discovering opinion spammer groups by network footprints," in *ECML-PKDD*, 2015.

[33] M. Ott, C. Cardie, and J. T. Hancock, "Negative deceptive opinion spam.," in *HLT-NAACL*, 2013.

[34] K. Shin, B. Hooi, and C. Faloutsos, "M-zoom: Fast dense-block detection in tensors with quality guarantees," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2016.

[35] K. Shin, B. Hooi, J. Kim, and C. Faloutsos, "D-cube: Dense-block detection in terabyte-scale tensors," in *WSDM*, ACM, 2017.

[36] M. Ott, C. Cardie, and J. Hancock, "Estimating the prevalence of deception in online review communities," in *WWW*, ACM, 2012.

[37] V. Sandulescu and M. Ester, "Detecting singleton review spammers using semantic similarity," in *WWW*, 2015.

[38] J. Piskorski, M. Sydow, and D. Weiss, "Exploring linguistic features for web spam detection: a preliminary study," in *Proceedings of the 4th international workshop on Adversarial information retrieval on the web*, 2008.

[39] J. Li, M. Ott, C. Cardie, and E. Hovy, "Towards a general rule for identifying deceptive opinion spam," in *ACL*, 2014.

[40] C. Harris, "Detecting deceptive opinion spam using human computation," in *Workshops at AAAI on Artificial Intelligence*, 2012.

[41] B. Hooi, N. Shah, A. Beutel, S. Günnemann, L. Akoglu, M. Kumar, D. Makhija, and C. Faloutsos, "Birdnest: Bayesian inference for ratings-fraud detection," in *SDM*, 2016.

[42] N. Shah, A. Beutel, B. Hooi, L. Akoglu, S. Gunnemann, D. Makhija, M. Kumar, and C. Faloutsos, "Edgecentric: Anomaly detection in edge-attributed networks," in *ICDMW*, 2016.

[43] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "Fraudar: Bounding graph fraud in the face of camouflage," in *KDD*, ACM, 2016.

[44] H. Li, G. Fei, S. Wang, B. Liu, W. Shao, A. Mukherjee, and J. Shao, "Bimodal distribution and co-bursting in review spam detection," in *Proceedings of the 26th International Con-*

ference on World Wide Web, pp. 1063–1072, International World Wide Web Conferences Steering Committee, 2017.

[45] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang, "Inferring strange behavior from connectivity pattern in social networks," in *PAKDD*, 2014.

[46] N. Shah, A. Beutel, B. Gallagher, and C. Faloutsos, "Spotting suspicious link behavior with fbox: An adversarial perspective," in *ICDM*, IEEE, 2014.

[47] B. A. Prakash, A. Sridharan, M. Seshadri, S. Machiraju, and C. Faloutsos, "Eigenspokes: Surprising patterns and scalable community chipping in large graphs," in *PAKDD*, 2010.

[48] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, "Combating web spam with trustrank," in *VLDB*, VLDB Endowment, 2004.

[49] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos, "Netprobe: a fast and scalable system for fraud detection in online auction networks," in *WWW*, ACM, 2007.

[50] E. Reiter and et al., "Choosing words in computer-generated weather forecasts," *Artificial Intelligence*, 2005.

[51] M. Makadia, "What are the advantage of natural language generation and its impact on business intelligence?, https://www.marutitech.com/advantages-of-natural-language-generation/, last access: 01/28/2019," 2018.

[52] A. K. Yadav and S. K. Borgohain, "Sentence generation from a bag of words using n-gram model," in *ICACCCT*, 2014.

[53] A. Graves, "Generating sequences with recurrent neural networks," *arXiv*, 2013.

[54] I. Sutskever and et al., "Generating text with recurrent neural networks," in *ICML*, 2011.

[55] S. Merity and et al., "Regularizing and optimizing lstm language models," *arXiv*, 2017.

[56] A. Xu and et al., "A new chatbot for customer service on social media," in *CHI*, 2017.

[57] L. Shang, Z. Lu, and H. Li, "Neural responding machine for short-text conversation," *arXiv*, 2015.

[58] A. Kannan and et al., "Smart reply: Automated response suggestion for email," in *KDD*, 2016.

[59] I. V. Serban and et al., "Hierarchical neural network generative models for movie dialogues," *CoRR, abs/1507.04808*, 2015.

[60] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *CVPR*, 2015.

[61] W. Fedus and et al., "Maskgan: Better text generation via filling in the _," *ICLR*, 2018.

[62] G. Melis and et al., "On the state of the art of evaluation in neural language models," *ICLR*, 2017.

[63] Z. C. Lipton and et al., "Capturing meaning in product reviews with character-level generative text models," *arXiv*, 2015.

[64] A. Radford and et al., "Learning to generate reviews and discovering sentiment," *arXiv*, 2017.

[65] D. Hovy, "The enemy in your own camp: How well can we detect statistically-generated fake reviews–an adversarial study," in *ACL*, 2016.

[66] M. Juuti and et al., "Stay on-topic: Generating context-specific fake restaurant reviews," in *ESORICS*, 2018.

[67] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *ACL*, 2018.

[68] D. Gerz and et al., "On the relation between linguistic typology and (limitations of) multilingual language modeling," in *EMNLP*, 2018.

[69] R. Sennrich and et al., "Improving neural machine translation models with monolingual data," *ACL*, 2015.

[70] S. Min and et al., "Question answering through transfer learning from large fine-grained supervision data," *arXiv*, 2017.

[71] H.-Q. Nguyen-Son and et al., "Identifying computer-generated text using statistical analysis," in *APSIPA ASC*, 2017.

[72] R. Aharoni and et al., "Automatic detection of machine translated text and translation quality estimation," in *ACL*, 2014.

[73] I. Goodfellow and et al., "Generative adversarial nets," in *NIPS*, 2014.

[74] A.-M. Popescu and O. Etzioni, "Extracting product features and opinions from reviews," in *Natural language processing and text mining*, Springer, 2007.

[75] S. Moghaddam and M. Ester, "Ilda: interdependent lda model for learning latent aspects and their ratings from online product reviews," in *SIGIR*, ACM, 2011.

[76] K. Liu, H. L. Xu, Y. Liu, and J. Zhao, "Opinion target extraction using partially-supervised word alignment model," in *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[77] L. Shu, B. Liu, H. Xu, and A. Kim, "Lifelong-rl: Lifelong relaxation labeling for separating entities and aspects in opinion targets," in *EMNLP*, vol. 2016, p. 225, NIH Public Access, 2016.

[78] H. Xu, B. Liu, L. Shu, and P. S. Yu, "Double embeddings and cnn-based sequence labeling for aspect extraction," *ACL*, 2018.

[79] S. e. a. Chen, "Driven answer generation for product-related questions in e-commerce," 2019.

[80] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *ICLR*, 2015.

[81] L. Dong, S. Huang, F. Wei, M. Lapata, M. Zhou, and K. Xu, "Learning to generate product reviews from attributes," in *EACL*, pp. 623–632, 2017.

[82] J. Ni and J. McAuley, "Personalized review generation by expanding phrases and attending on aspect-aware representations," in *ACL*, pp. 706–711, 2018.

[83] H. Zang and X. Wan, "Towards automatic generation of product reviews from aspect-sentiment scores," in *Proceedings of the 10th International Conference on Natural Language Generation*, pp. 168–177, 2017.

[84] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *SIGIR*, ACM, 2015.

[85] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining*, 2006.

[86] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[87] Z. S. Harris, "Distributional structure," *Word*, 1954.

[88] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*, ACM, 2014.

[89] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*, ACM, 2015.

[90] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*, ACM, 2016.

[91] P. Grey, "How many products does amazon sell?, https://export-x.com, last access: 01/10/2017," 2015.

[92] N. Hu, P. A. Pavlou, and J. Zhang, "Can online reviews reveal a product's true quality?: empirical findings and analytical modeling of online word-of-mouth communication," in *ICEC*, 2006.

[93] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, 2015.

[94] B. Viswanath, M. A. Bashir, M. B. Zafar, S. Bouget, S. Guha, K. P. Gummadi, A. Kate, and A. Mislove, "Strength in numbers: Robust tamper detection in crowd computations,"

in *Proceedings of the 2015 ACM on Conference on Online Social Networks*, pp. 113–124, ACM, 2015.

[95] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw, "Detecting product review spammers using rating behaviors," in *CIKM*, ACM, 2010.

[96] S. Li, J. Caverlee, W. Niu, and P. Kaghazgaran, "Crowdsourced app review manipulation," in *SIGIR*, 2017.

[97] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 1997.

[98] S. Ahn, H. Choi, T. Pärnamaa, and Y. Bengio, "A neural knowledge language model," *arXiv preprint arXiv:1608.00318*, 2016.

[99] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, 2014.

[100] F. Chollet, *Deep learning with Python*. Manning Publications Co., 2017.

[101] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[102] L. Wu, X. Hu, F. Morstatter, and H. Liu, "Adaptive spammer detection with sparse group modeling," in *ICWSM*, 2017.

[103] S. Li and et al., "Crowdsourced app review manipulation," in *SIGIR*, 2017.

[104] S. Merity and et al., "An analysis of neural language modeling at multiple scales," *arXiv*, 2018.

[105] T. Mikolov and et al., "Recurrent neural network based language model," in *ISCA*, 2010.

[106] T. Mikolov and et al., "Empirical evaluation and combination of advanced language modeling techniques,"

[107] E. Arisoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Deep neural network language models," in *NAACL-HLT*, ACL, 2012.

[108] H. Inan and et al., "Tying word vectors and word classifiers: A loss framework for language modeling," *arXiv*, 2016.

[109] J. Yosinski and et al., "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, 2014.

[110] M. Peters and et al., "Semi-supervised sequence tagging with bidirectional language models," *arXiv*, 2017.

[111] M. E. e. a. Peters, "Deep contextualized word representations," *arXiv*, 2018.

[112] W. H. Gomaa and A. A. Fahmy, "A survey of text similarity approaches," *International Journal of Computer Applications*, 2013.

[113] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining.," in *Lrec*, 2010.

[114] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, 2008.

[115] O. Vinyals and Q. Le, "A neural conversational model," *ICML*, 2015.

[116] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *International conference on machine learning*, 2015.

[117] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[118] J. Gao, D. He, X. Tan, T. Qin, L. Wang, and T.-Y. Liu, "Representation degeneration problem in training natural language generation models," *ICLR*, 2019.

[119] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, A.-S. Mohammad, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq, *et al.*, "Semeval-2016 task 5: Aspect based sentiment analysis," in *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pp. 19–30, 2016.

[120] A. A. Alemi and P. Ginsparg, "Text segmentation based on semantic word embeddings," *arXiv preprint arXiv:1503.05543*, 2015.

[121] D. Beeferman, A. Berger, and J. Lafferty, "Statistical models for text segmentation," *Machine learning*, 1999.

[122] P. M. McCarthy and S. Jarvis, "Mtld, vocd-d, and hd-d: A validation study of sophisticated approaches to lexical diversity assessment," *Behavior research methods*, vol. 42, no. 2, pp. 381–392, 2010.